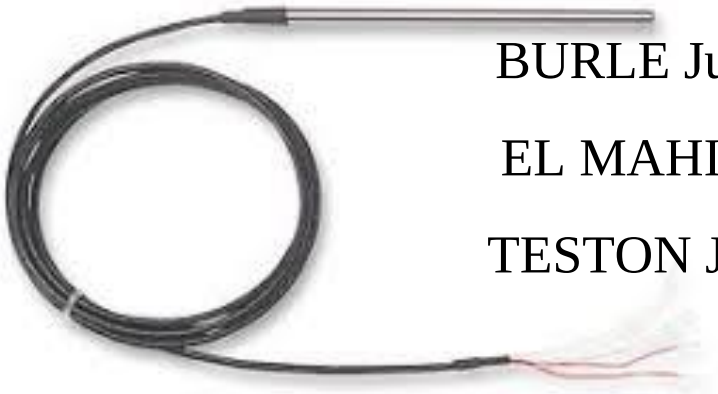
REVUE DE PROJET N°2

Projet en commun avec :



BURLE Julien (EC1)

EL MAHI Mehdi (IR)

TESTON Julien (EC2)

Professeurs référents :

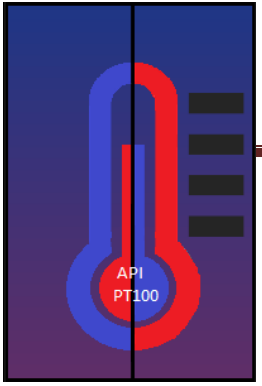
M.HORTOLLAND / M.DEFRANCE

Le commanditaire:

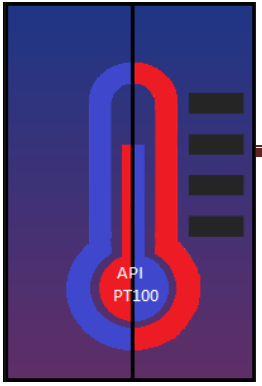
API / M.BUNEL

Lycée Alphonse Benoît

Année 2018-2019



Sommaire



1. Présentation de l'entreprise

API est une SARL (société à responsabilité limitée) active depuis 5 ans.
Implantée à VEDENE (84270) au 402 Avenue des Lacs.

Elle est spécialisée dans le secteur de la conception d'ensemble et assemblage sur site industriel d'équipements de contrôle des processus industriels. Son effectif varie de 3 à 5 salariés.
Rudy Sylvestre est gérant de la société API.
La société API intervient dans différents types d'industries et secteurs d'activités comme :

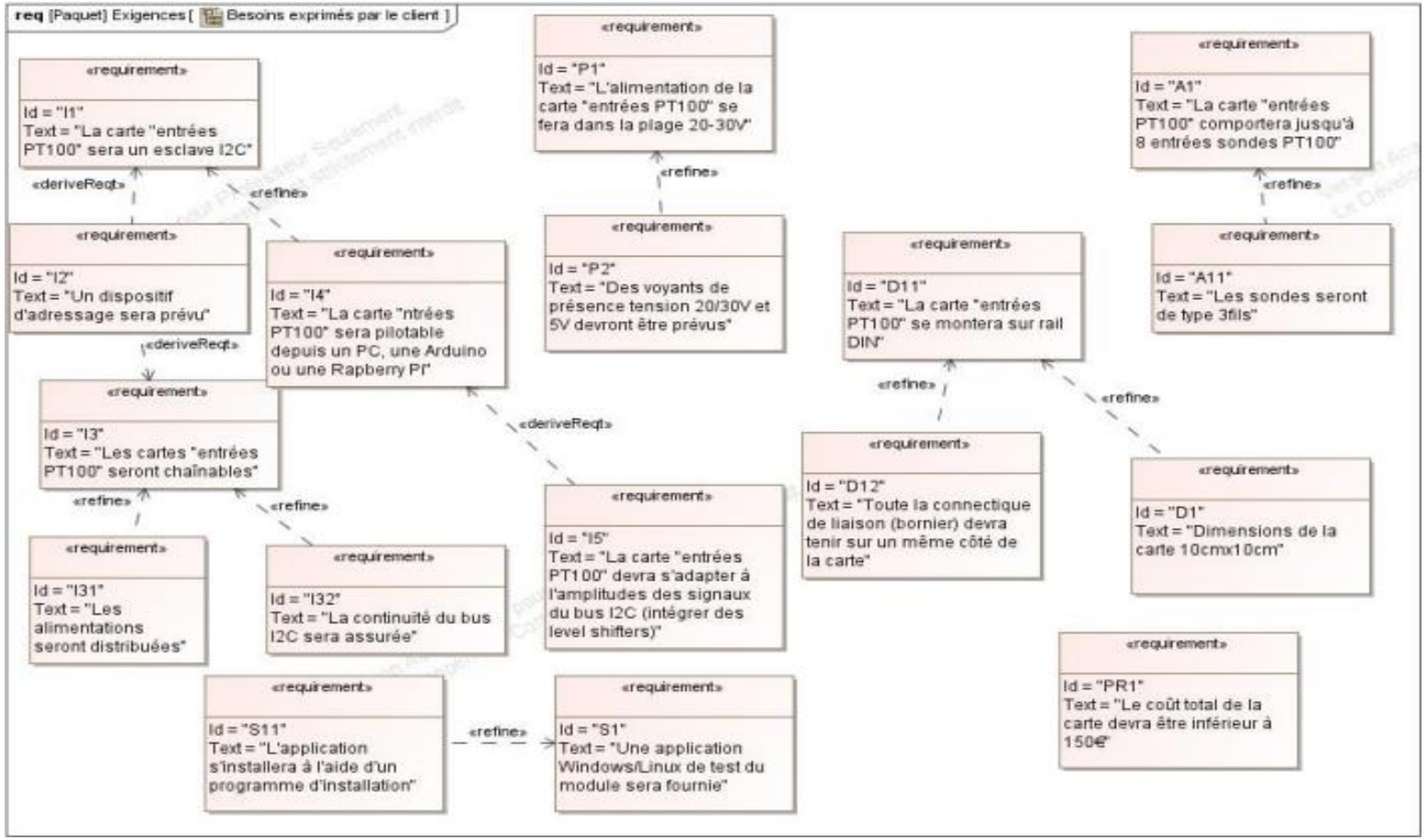
- L'industrie du verre
- L'agro-alimentaire
- L'industrie du papier
- Les carrières
- Les machines spéciales
- TPE et artisans

Elle possède également une activité axée sur la production et la gestion des énergies. Dans ces domaines, API propose notamment les services suivants :

- Audit des réseaux d'énergie (air et électrique).
- Mesures, historisations, bilan des consommations.
- Préconisation d'économie d'énergie (variation de vitesse, régulation ...).
- Automatisation des procédés de gestion, Domotique, Immotique.
- Etude et calcul d'installation photovoltaïque.

2. Les exigences

2.1. Contraintes



Contraintes financières : Budget estimé de 200 à 300€
L'entreprise API participera au financement du projet.

Contraintes de développement): La spécification, conception et codage seront modélisés.
Contraintes qualité (conformité, délais, …) : Maintenable, maniable (ergonomie)

Contraintes de fiabilité, sécurité: Les accès logiciels seront sécurisés.

2.2. Ressources à disposition

Matériels :

- **PC Windows/Linux**
- **Cartes Raspberry Pi**
- **Composants et matériel de câblage**
- **Adaptateur USB↔I2C**

- **Analyseur logique**
- **Platine d'essai type Labdec**
- **2 cartes 8 entrées analogiques développées lors de la session du BTS 2018**

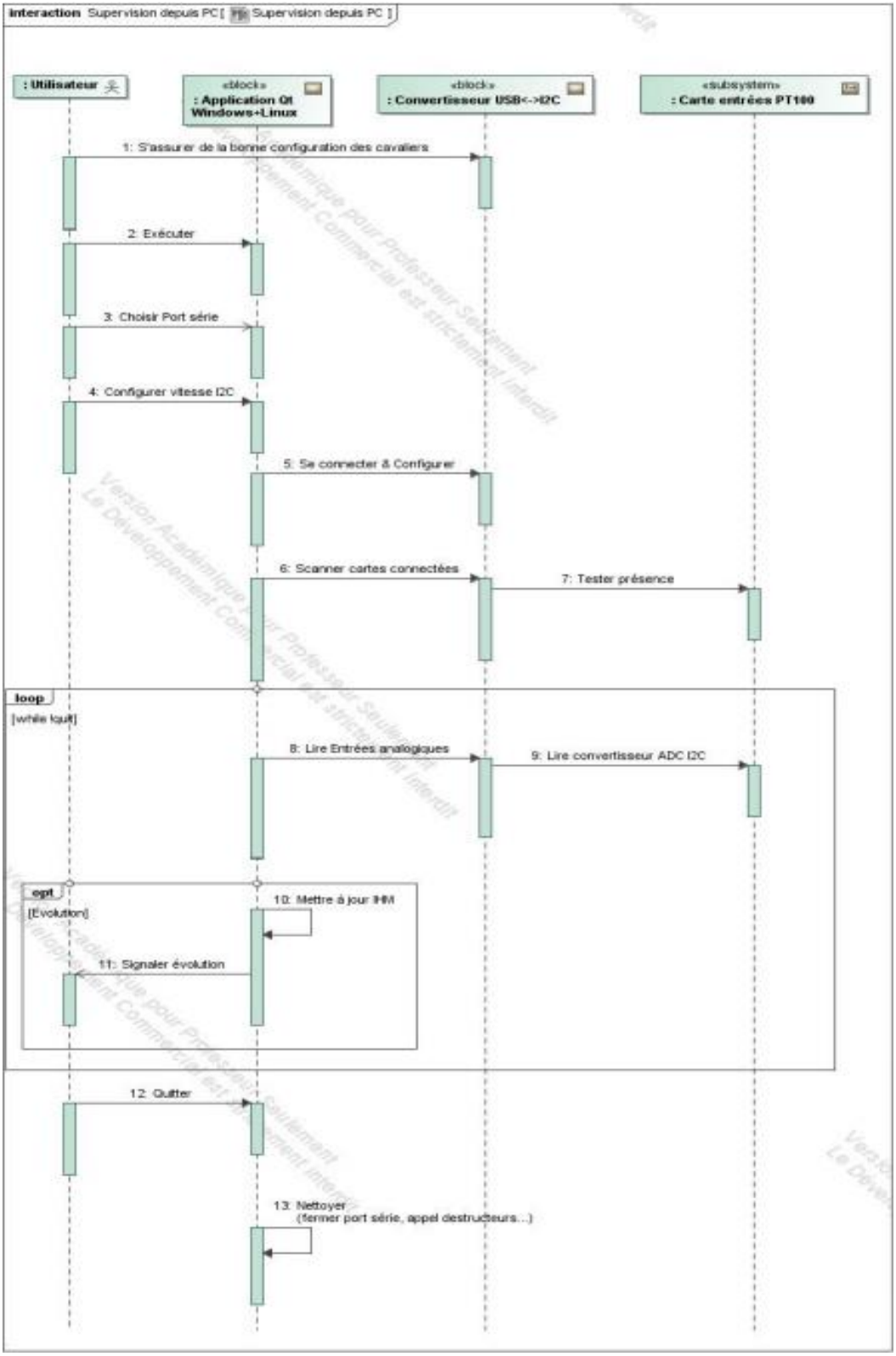
Logiciels :

- **Logiciel de modélisation SysML/UML : MagicDraw v7.02**
- **Logiciels de conception électronique : KiCad 5 (routage, simulation éventuelle), Proteus (simulation éventuelle), TI Webenchfilters (simulation éventuelle).**
- **Logiciel de conception électronique Fritzing pour illustrer le prototypage rapide •**
- **Système d'exploitation Linux (Raspbian)**
- **Framework Qt/C++**

Documents :

- **Site de la section BTS SN mettant à disposition les différentes documentations.**

3. Cas d'utilisation

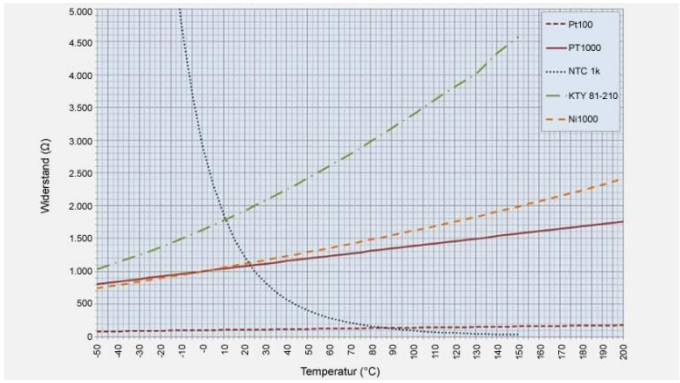


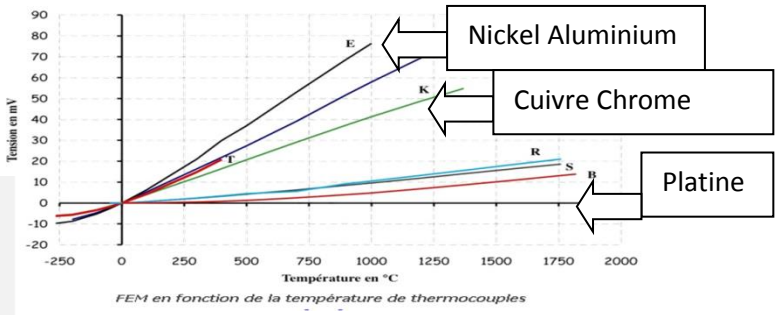
4. Le Capteur

4.1. Qu'est ce qu'une PT100

Une sonde PT100 est un type de capteur de température aussi appelé RTD (Resistance Temperature Detector, détecteur de température à résistance) qui est fabriqué à partir de platine pour une meilleure précision. La sonde PT100 a une résistance de 100 ohms à 0 °C, et il est de loin le capteur le plus utilisé. Le capteur PT500 a une résistance de 500 ohms à 0 °C, et le capteur PT1000 a une résistance de 1000 ohms à 0 °C. Normalement, ces capteurs sont équipés d'une gaine de protection ou de montage pour former une sonde de température, et ceux-ci sont couramment appelés des PRT (thermomètre à résistance de platine).

- Températures de -200 °C à 850 °C
- Courbe caractéristique quasi linéaire
- Précision élevée





FEM en fonction de la température de thermocouples

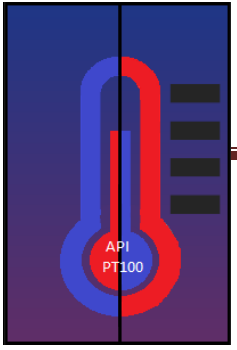
La précision peut être influencée par d'autres paramètres tels que :

- Le diamètre du câble
- La classe de la sonde
- Le mode de connexion
- Le mode d'acquisition
- Le couplage thermique
- L'auto échauffement

4.2. Pourquoi la PT100 3fils

Il existe trois sortes de sondes PT100 :

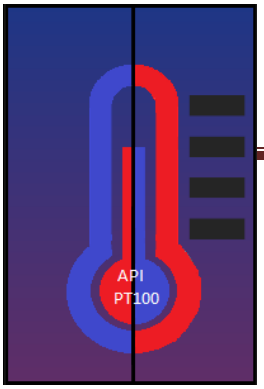
- Sonde PT100 2fils
  La plus simple : les mesures sont précises à courtes distances et il faut prévoir une marge d'erreur de 0.4°C par mètre.
- Sonde PT100 3fils
  Précision suffisante pour les grandes distances de câblage avec une marge d'erreur de 0.4°C pour 100m
- Sonde PT100 4fils
  La meilleure précision, la mesure se faisant au niveau actif de la sonde et n'est pas influencée par la longueur du câblage.



5. Le projet

5.1. Tâches à accomplir par les étudiants

Étudiant 1 **IR**	*Liste des tâches assurées par l'étudiant* • Mettre en œuvre l'adaptateur USB↔I2C • Etudier les datasheets des composants I2C • Câbler une plaquette d'essai disposant des composants I2C présents dans la carte « Entrées PT100 ». • Concevoir/Coder/Tester des classes C++ d'accès aux composants I2C (ADC) via l'adaptateur USB↔I2C • Concevoir/Coder/Tester une application Qt Windows+Linux de pilotage de cartes « entrées PT100 » • Créer un programme d'installation de l'application sur Windows (et éventuellement un paquet type rpm pour Linux) • Assurer la gestion de version logicielle (Git+Bitbucket) • Rédiger un manuel de démarrage rapide pour l'installation et l'utilisation de l'application	**Installation** : Framework Qt (x64) **Mise en œuvre :** Framework Qt/C++, ADC sur bus I2C **Configuration** : ..... **Réalisation** : Concevoir une application de pilotage de la carte « entrées PT100 » **Documentation** : Guide d'installation, manuel utilisateur, dossier de développement
Étudiant 2 **EC 1**	*Liste des tâches assurées par l'étudiant* • Analyser le schéma de la carte 8 entrées analogiques du projet de BTS 2018. • Mettre en service cette carte. • Analyser la notice d'application mise à votre disposition (DS00687C Microchip). Réaliser un câblage rapide de cette structure *(commander les composants si nécessaire)* et la substituer à une partie de la carte API IO AN du BTS 2018. Effectuer tous les essais nécessaires pour valider ou non la structure, ou lui apporter des modifications. Un logiciel de simulation pourra éventuellement être utilisé pour tester/illustrer le fonctionnement de l'étage d'alimentation et de conditionnement du signal issu du capteur. • Effectuer des recherches poussées d'approvisionnement de tous les composants pour dresser la nomenclature de la carte la moins cher possible. • Effectuer la saisie du schéma et le routage de cette carte. Produire les fichiers Gerber afin que la fabrication du PCB soit sous-traitée. • Câbler le PCB de la carte et effectuer les essais. • Adapter l'application graphique (Qt) de démonstration de la version 2018 sur Raspberry Pi, afin qu'elle permette d'afficher la température des sondes PT100.	**Installation** : Mise en service (initialisation/configuration) d'une Raspberry Pi : - communication I2C avec librairie BCM2835, - Qt Creator. **Mise en œuvre :** Valider par prototypage rapide une structure permettant l'acquisition de signaux issus de sondes PT100 à 3 broches. Proposer des modifications le cas échéant. *S'il s'avère que l'étudiant EC2 a une structure plus performante, vous utiliserez cette dernière.* **Réalisation :** Suite aux essais préalables, finaliser le schéma structurel de la carte d'acquisition, en réutilisant (et modifiant) le projet KiCad de 2018. Concevoir un circuit imprimé devant être fabriqué industriellement. **Documentation :** Schéma de câblage rapide (Fritzing). Documents de fabrication des cartes (KiCad). Ces documents devront avoir un niveau de qualité permettant une fabrication industrielle du circuit imprimé. Bibliothèque d'acquisition des mesures.
Étudiant 3 **EC 2**	*Liste des tâches assurées par l'étudiant* • Analyser le schéma de la carte 8 entrées analogiques du projet de BTS 2018. • Mettre en service cette carte. • Analyser la notice d'application mise à votre disposition (TIDU969 Texas Instrument). Réaliser un câblage rapide de cette structure *(commander les composants si nécessaire)* et la substituer à une partie de la carte API IO	**Installation** : Mise en service (initialisation/configuration) d'une Raspberry Pi : - communication I2C avec librairie BCM2835, - Qt Creator. **Mise en œuvre :** Valider par prototypage rapide une structure permettant l'acquisition de signaux issus de sondes PT100 à 3 broches. Proposer des modifications le cas échéant.
	AN du BTS 2018. Effectuer tous les essais nécessaires pour valider ou non la structure, ou lui apporter des modifications. Un logiciel de simulation pourra éventuellement être utilisé pour tester/illustrer le fonctionnement de l'étage d'alimentation et de conditionnement du signal issu du capteur. • Effectuer des recherches poussées d'approvisionnement de tous les composants pour dresser la nomenclature de la carte la moins cher possible. • Effectuer la saisie du schéma et le routage de cette carte. Produire les fichiers Gerber afin que la fabrication du PCB soit sous-traitée. • Câbler le PCB de la carte et effectuer les essais. • Adapter l'application graphique (Qt) de démonstration de la version 2018 sur Raspberry Pi, afin qu'elle permette d'afficher la température des sondes PT100.	*S'il s'avère que l'étudiant EC1 a une structure plus performante, vous utiliserez cette dernière.* **Réalisation :** Suite aux essais préalables, finaliser le schéma structurel de la carte d'acquisition, en réutilisant (et modifiant) le projet KiCad de 2018. Concevoir un circuit imprimé devant être fabriqué industriellement. **Documentation :** Schéma de câblage rapide (Fritzing). Documents de fabrication des cartes (KiCad). Ces documents devront avoir un niveau de qualité permettant une fabrication industrielle du circuit imprimé. Bibliothèque d'acquisition des mesures.



6. EC2 TESTON Julien

Le projet est réalisé par trois étudiants, deux EC et un IR.
En ce qui concerne, ma partie j'ai travaillé sur la suite d'un projet de 2018 :

- Analyse de la carte existante
- Analyse du programme sous QT
- Analyse du schéma structurel
- Prise de connaissance de toutes les documentations
- Familiarisation avec le capteur PT100 qui m'était jusqu'à là inconnu

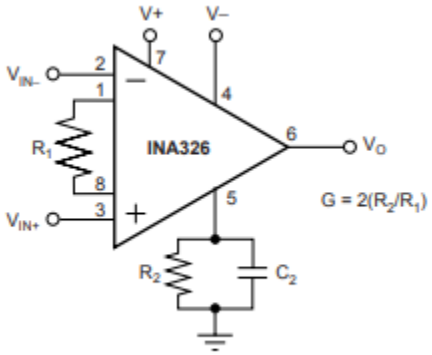
Pour ce qui concerne le reste du projet, j'ai travaillé avec Julien Burle, qui lui s'est occupé de faire la 2nde solution (Microchip**)**.

Le contrat avec API nous oblige à choisir une des deux solutions afin de nous focaliser sur cette dernière.
Il y a donc un contrat pour 2 EC.

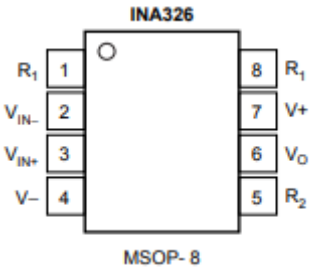
Installation
Librairies KiCad
Installation KiCad
Mise en œuvre
Schémas structurels de la carte de test et de la carte finale.
Routage des schémas.
Remise en marche de la carte existante (API AN 2018).
Câblage de la solution Texas instrument

6.1. Composants utilisé pour la Solution Texas instrument

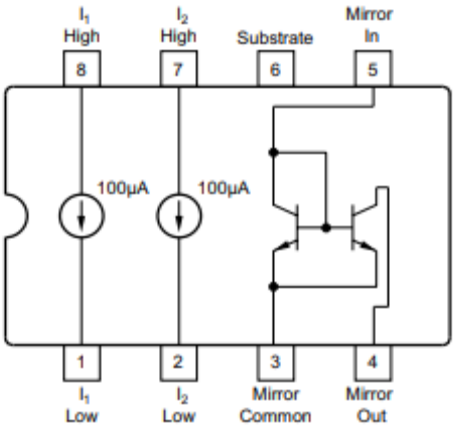
Les INA326 sont des amplificateurs d'instrumentation de haute précision. Performances, marge d'erreur très basse. Ces caractéristiques les rendent aptes à des applications allant de la polyvalence à la haute précision.

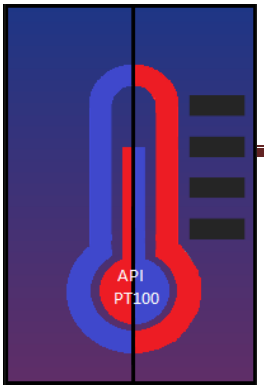


Et offrent une excellente stabilité à long terme et un très faible bruit pendant toute la durée de vie du produit.



Le REF200 combine trois modules sur une même puce monolithique: deux sources de courant de 100 µA et un miroir de courant. Les sections sont isolées électriquement, ce qui les rend complètement indépendantes. De plus, comme les sources de courant sont des dispositifs à deux terminaux, elles peuvent être utilisées comme des puits de courant. Les performances de chaque section sont mesurées individuellement et ajustées au laser afin d'obtenir une grande précision à moindre coût. Les sections peuvent être attachées avec des courants de 50 µA, 100 µA, 200 µA, 300 µA ou 400 µA. les circuits externes peuvent obtenir pratiquement n'importe quel courant.





6.2. Solution proposé (Texas Instruments)

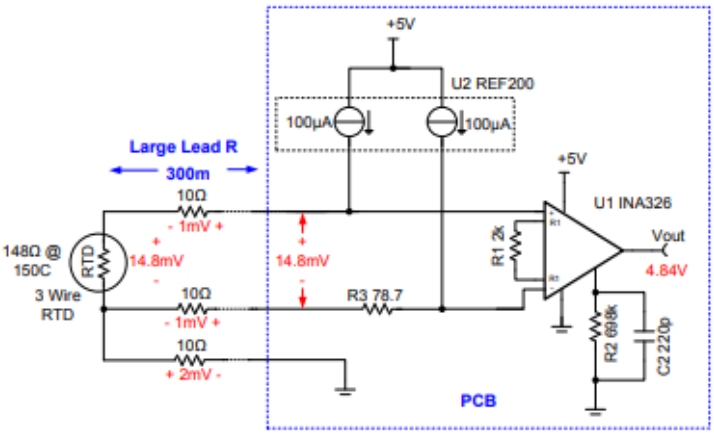
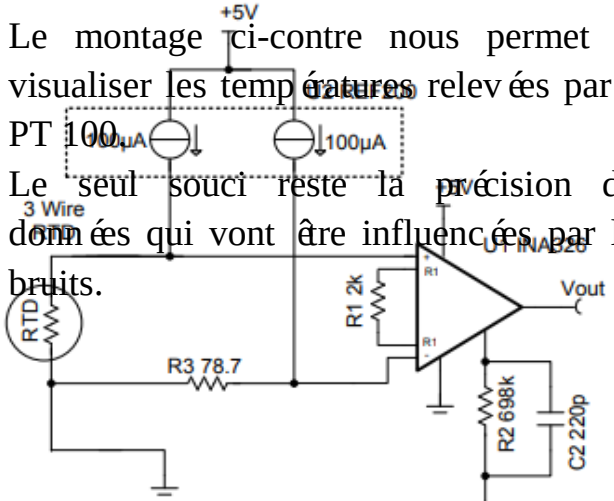


Figure 4: Three wire RTD configuration cancels lead resistance

La figure 4 ci-contre montre que la configuration RTD à trois fils peut être utilisée pour annuler la résistance de fil. Notez que la résistance dans chaque conducteur doit être égale pour annuler l'erreur. En outre, les deux sources de courant dans le REF200 ont besoin d'être égales.

Le montage ci-contre nous permet de visualiser les températures relevées par la PT 100.

Le seul souci reste la précision des données qui vont être influencées par les bruits.



On a donc décidé d'intégrer un filtre au montage pour éviter le développement de bruit. Le filtre montré sur la figure 5 est utile pour atténuer le captage de bruit en mode commun.

Les condensateurs peuvent être mis sur le circuit car la mesure de la température étant une variable qui ne change pas très rapidement, ils ont le temps de faire leur cycle charge/décharge

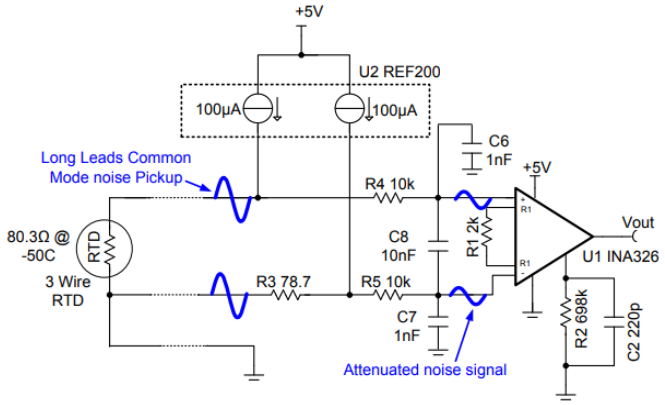


Figure 5: Common Mode and Differential Noise filter

6.3. Mise en situation

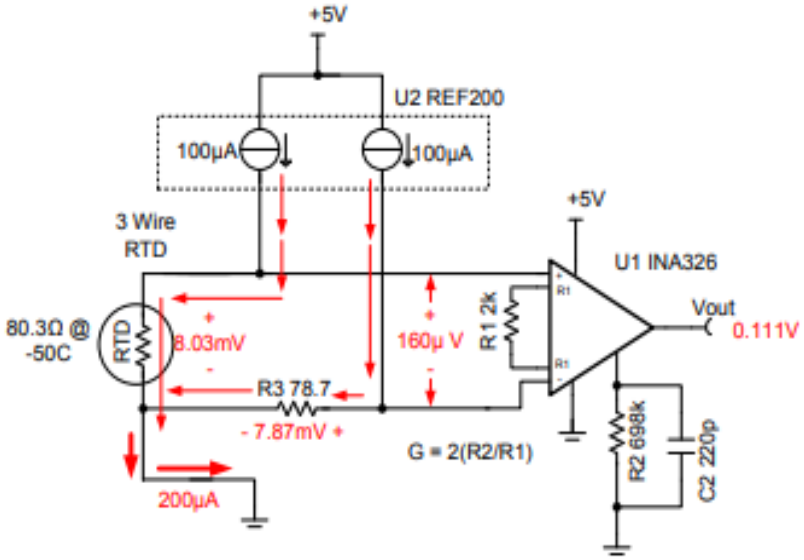


Figure 2: RTD Amplifier with Minimum Output Condition

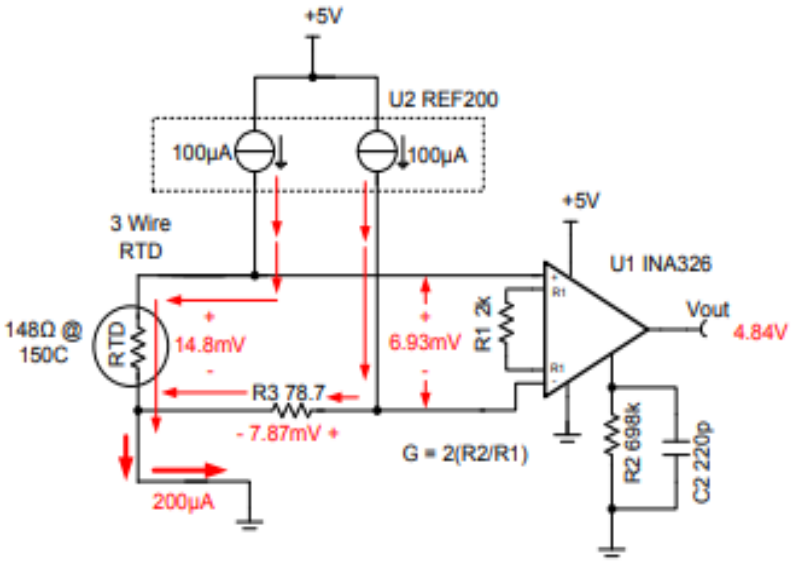


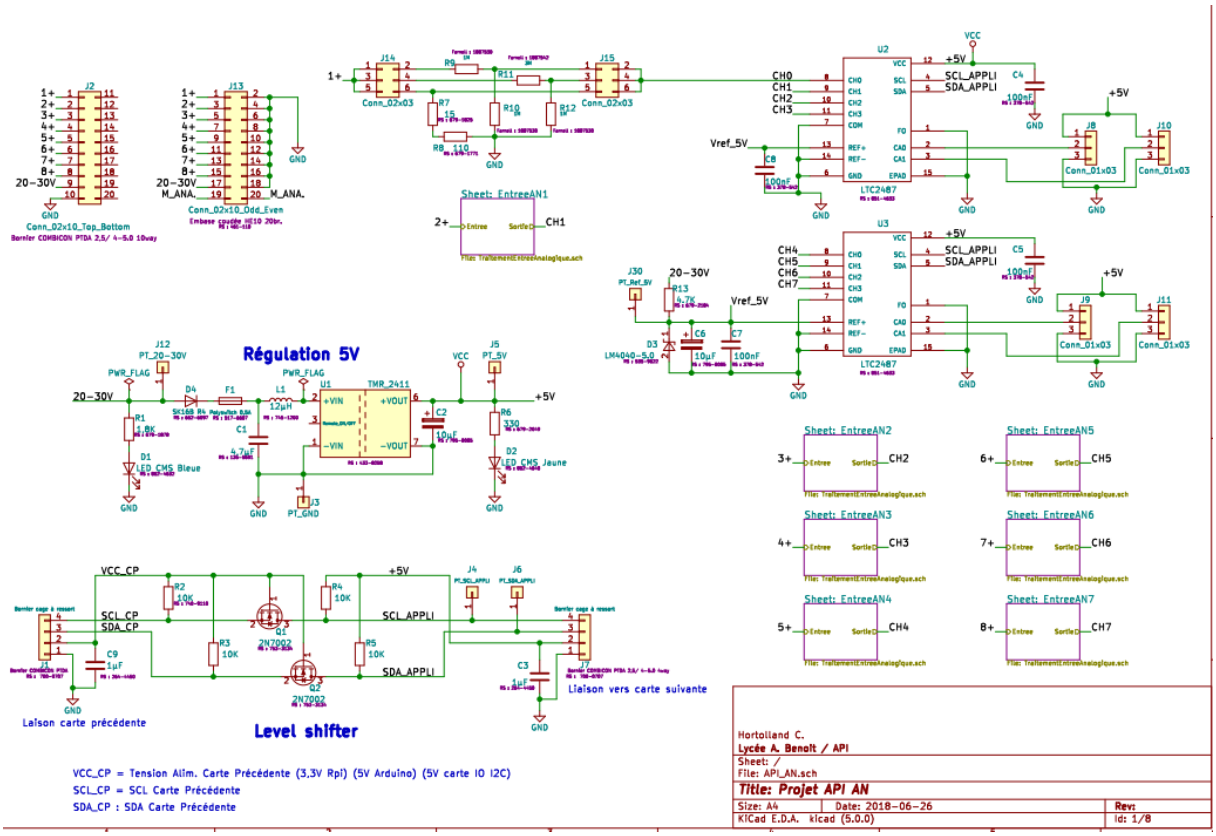
Figure 3: RTD Amplifier with Maximum Output Condition

La figure 2 et la figure 3 représentent le schéma de l'amplificateur RTD pour une sortie minimale et maximale.
Elles correspondent pour le moment à la température demandée par API.
La résistance de RTD est de 80,3Ω (pour -50 °C)
Et la tension qui la traverse est de 8,03 mV (VRTD = (100 µA)).
Notez que R3 développe une chute de tension qui s'oppose à la chute de RTD. La chute sur R3 est utilisée pour déplacer la tension différentielle d'entrée des amplificateurs à un niveau minimum. La sortie correspond à l'entrée différentielle multipliée par le gain (Vout = 698 x 160 µV = 0.111V). À 150 °C, la résistance RTD est de 148 Ω et la tension qui la traverse est de 14,8 mV (VRTD = (100µA) (148Ω)). Ceci produit une entrée différentielle de 6,93 mV et une tension de sortie de 4,84V (Vout = 698 x 6,93mV = 4,84V, voir figure 3).

6.4. Mise en œuvre

- Création des divers fichiers pour le compte rendu
  - ✓ Création du document Word et de la présentation
  - ✓ Création du logo
  - ✓ Création du Gantt

- Schémas structurels de la carte de test 2018 et de la carte 2019.



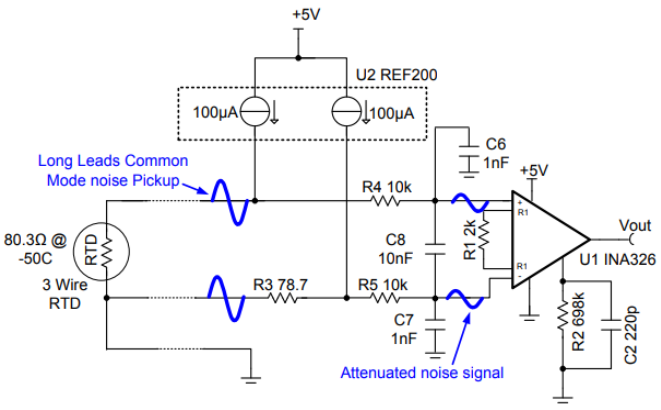
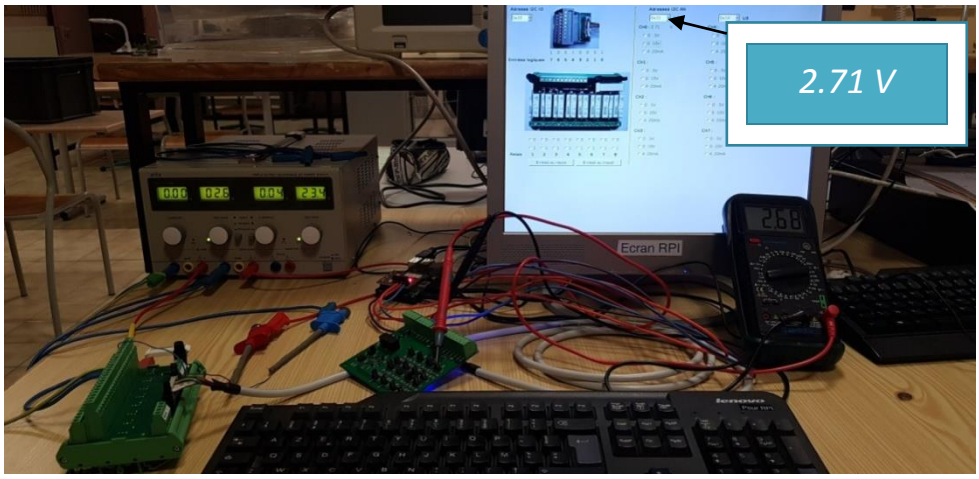


Figure 5: Common Mode and Differential Noise filter

6.5. Remise en marche de la carte existante.

Pour les tests des deux cartes, j'ai injecté une tension variant entre 1 et 5 Volts afin de voir ce que je recevais le logiciel. Pour valider les données j'ai utilisé un Voltmètre, il c'est avéré que le générateur ne donne pas la tension exacte :

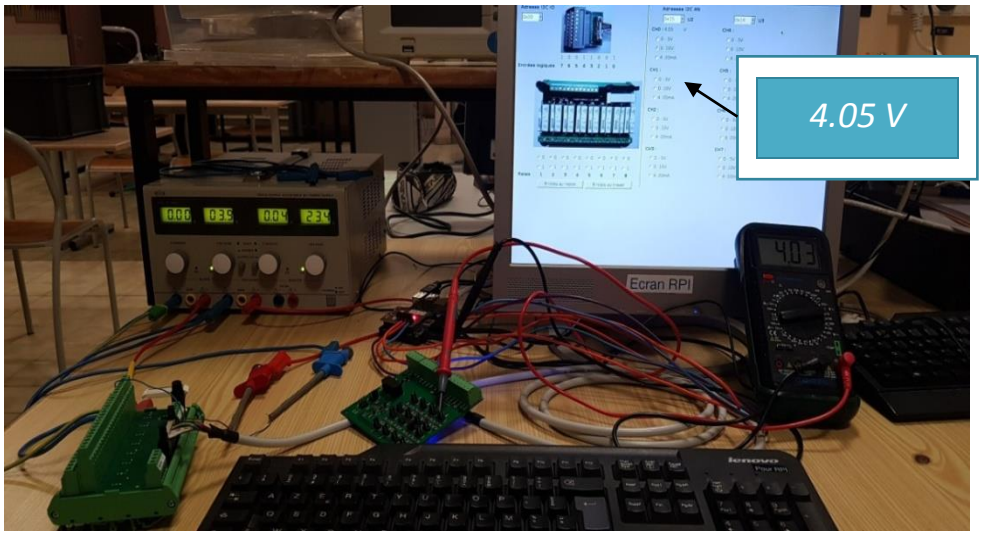


Carte 1, Test 1 :

On peut constater que la tension injectée de 2.6 V et bien retrouvée au niveau du multimètre (2.68 V) et au niveau de l'application.

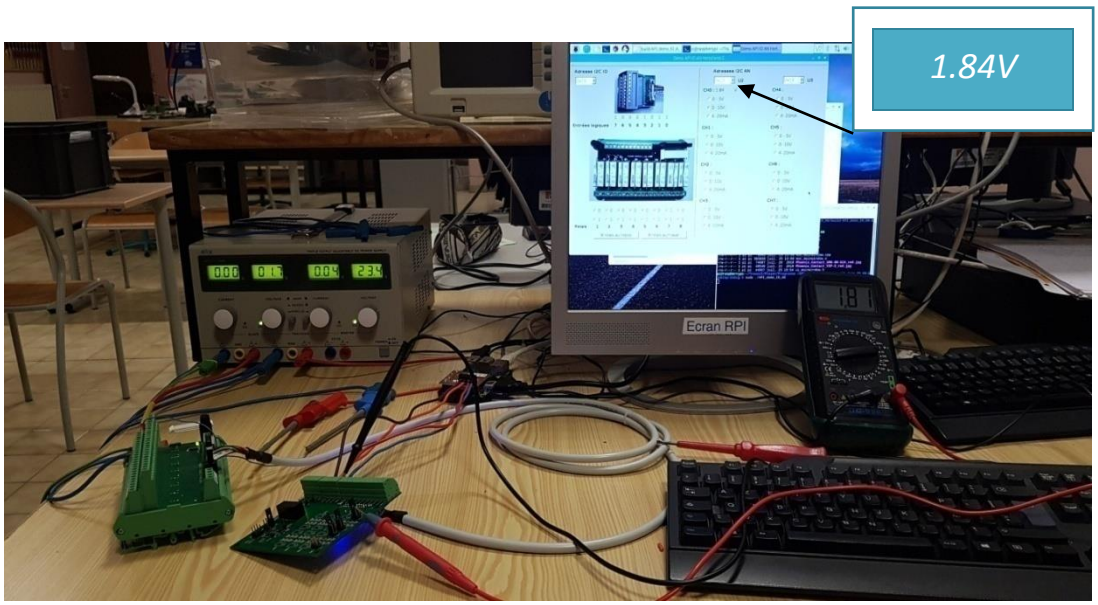
Carte 1, Test 2 :

A nouveau, on peut constater que la tension de 3.9 V injectée est retrouvée sur le multimètre (4.03) ainsi que sur le logiciel.



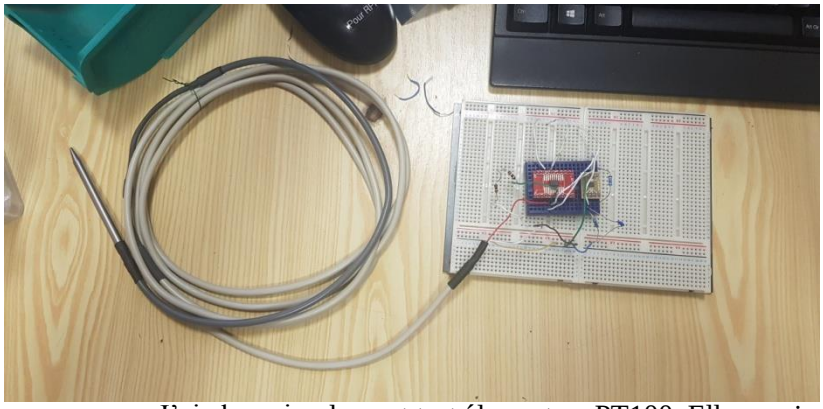
Les même tests ont été effectués sur la 2nd carte :

Test 1
Signal injecté : 2.9V Multimètre : 2.97V
Application : 3V



Test 2
Signal injecté : 1.7V
Multimètre : 1.81V
Application : 1.84V

6.6. Câblage de la solution Texas instrument

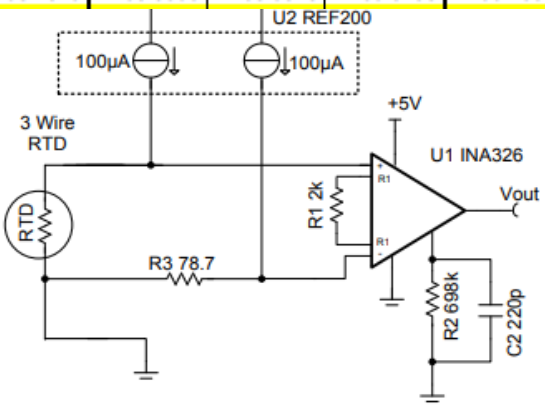


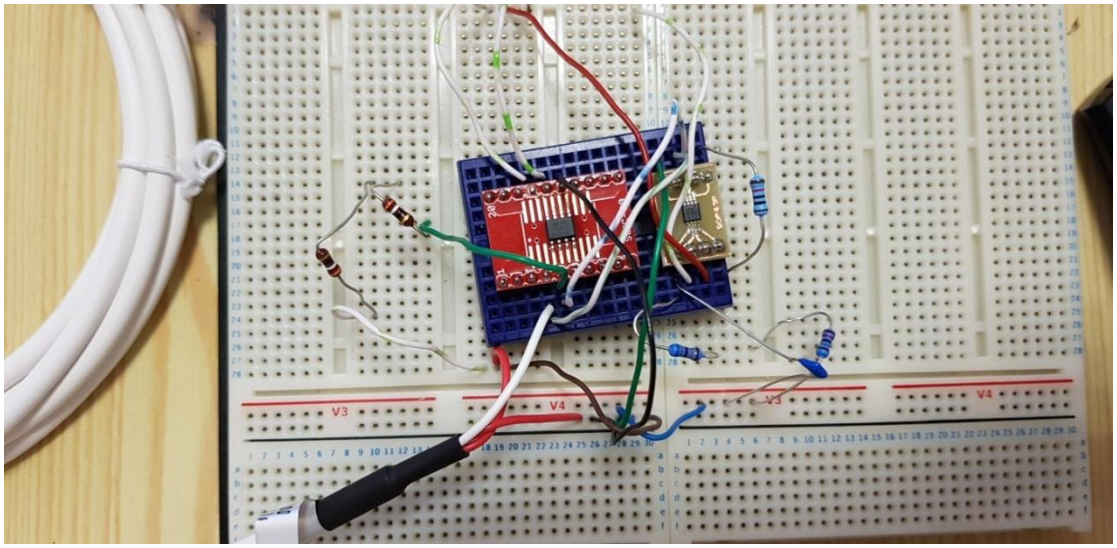
J'ai reproduit le montage avec le REF200 ainsi que l'INA326

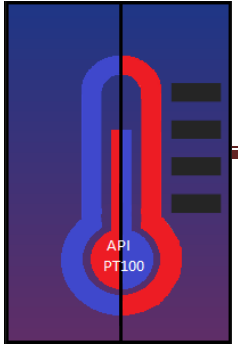
J'ai eu un souci avec la PT100 qui ne donnait pas de valeur cohérente lors des tests.
Les valeurs relevées étaient de 25-40 Ω ou encore 7Ω, ce qui correspond à une température négative.

J'ai alors simplement testé les autres PT100. Elles avaient une valeur plus proche du tableau de valeur (voir ci-dessous), j'ai donc remplacé la PT100 défectueuse.

0	100.0000	100.3907	100.7814	101.1719	101.5623	101.9526	102.3427	102.7328	103.1227	103.5125	0
10	103.9022	104.2918	104.6813	105.0706	105.4599	105.8490	106.2380	106.6269	107.0156	107.4043	10
20	107.7928	108.1813	108.5696	108.9578	109.3458	109.7338	110.1216	110.5094	110.8970	111.2845	20







6.7. Validation du Câblage

Pour la partie suivante du projet, j'ai mis en œuvre le câblage en breadboard afin de valider ce dernier.

J'ai rencontré les difficultés suivantes :

- Tout d'abord une erreur dans la résistance R1 (10 kΩ) a mis en place un Gain trop faible, ce qui fait que la plage de tension n'est pas respectée d'où des températures mesurées beaucoup trop élevées).
G=10
Soit 1.09/10=0.109
$0.109+(78.7*100.10^{-6})=0.11687$
$0.11687/100.10^{-6}=1168\Omega$
Donc une valeur hors norme pour une salle de classe

J'ai donc changé la résistance R1 par une résistance de 698kΩ mais le gain s'est retrouvé beaucoup trop fort. La plage de tension n'est là encore pas respectée pour les températures demandées par l'entreprise (-50 -> 150 °C). Les valeurs obtenues n'étaient donc pas normales (94Ω) ce qui correspond a une température négative … pour la salle de classe !
G=698
Soit $1.09/698=1.561.10^{-3}$
$1.561.10^{-3}+ (78.7*100.10^{-6})=9.4316.10^{-3}$
$9.4316.10^{-3}/100.10^{-6}=94\Omega$
Donc une valeur négative impossible pour une salle de classe

Ce problème était dû au gain de l'ampli qui était trop élevé. A l'aide du professeur de physique, j'ai recommencé tous mes calculs et nous avons mis en place un nouveau Gain de 348 en modifiant R1 par 698kΩ et R2 par 4kΩ.
G=348
Soit $1.09/348=3.1321.10^{-3}$
$3.1321^{-3}+ (78.7*100.10^{-6})=0.011$
$0.011/100.10^{-6}=110.02\Omega$
Ce qui correspond à une salle de classe ~25 °C

Mais les résistances CMS étant déjà en commande, j'ai dû m'adapter et remodifier les résistances, j'ai recalculé et modifié les résistances R1 et R2 par 348kΩ et 2kΩ afin de garder le Gain de 348.

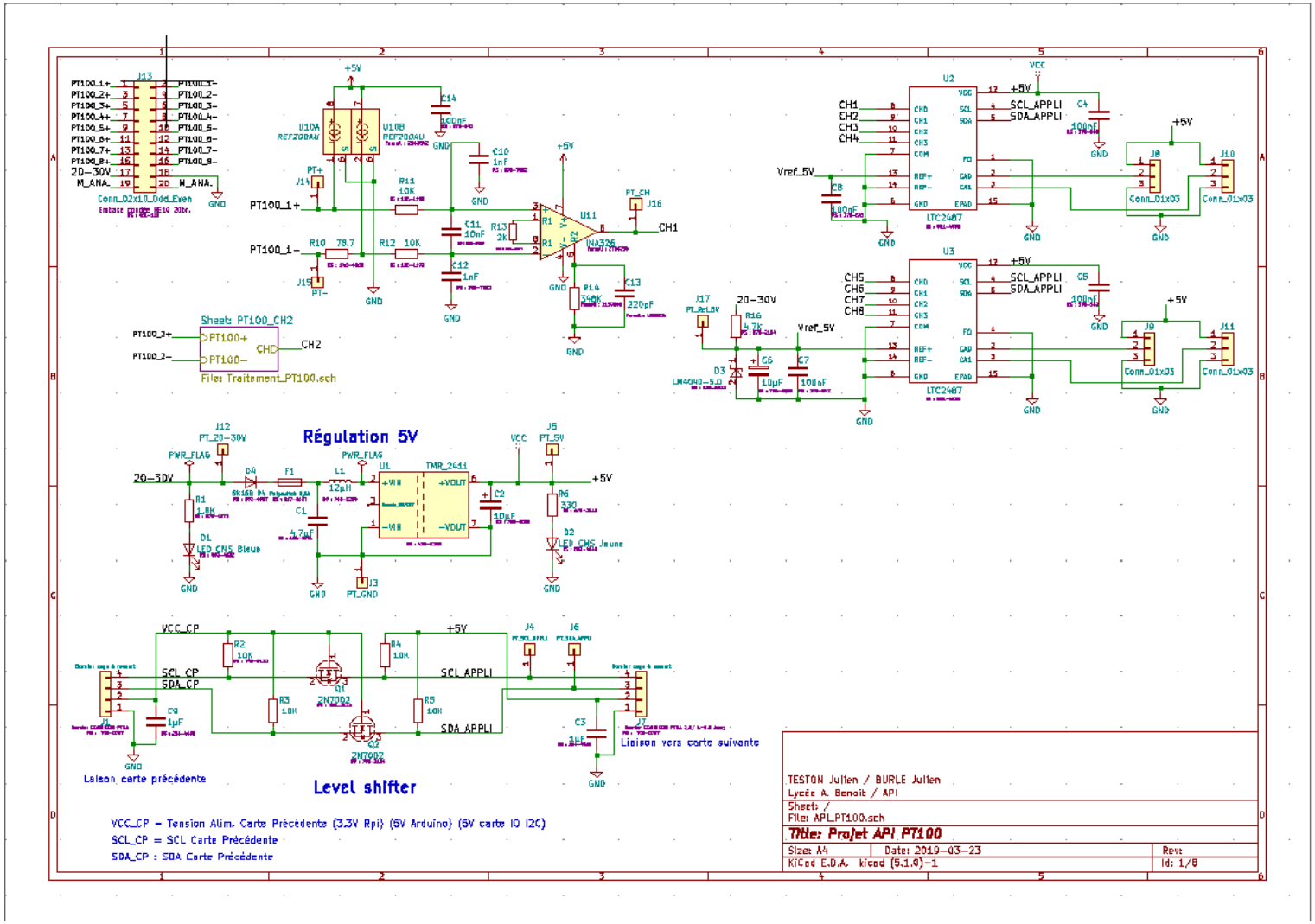
- J'ai alors refait tous les tests et me suis aperçu d'une différence de 0.2mV entre la tension mesurée sur le module et celle mesurée à la sortie du Breadboard.

Après avoir effectué plusieurs mesures pour trouver d'où vient le problème j'ai trouvé deux alternatives : soit le souci vient du Breadboard et il va se régler quand la carte finale arrivera car le montage en l'air peut engendrer des résistances différentes (du à la longueur des fils) que le montage CMS annulera, soit il faudra prendre en compte cette différence dans le programme.

6.8. Schémas via KiCad

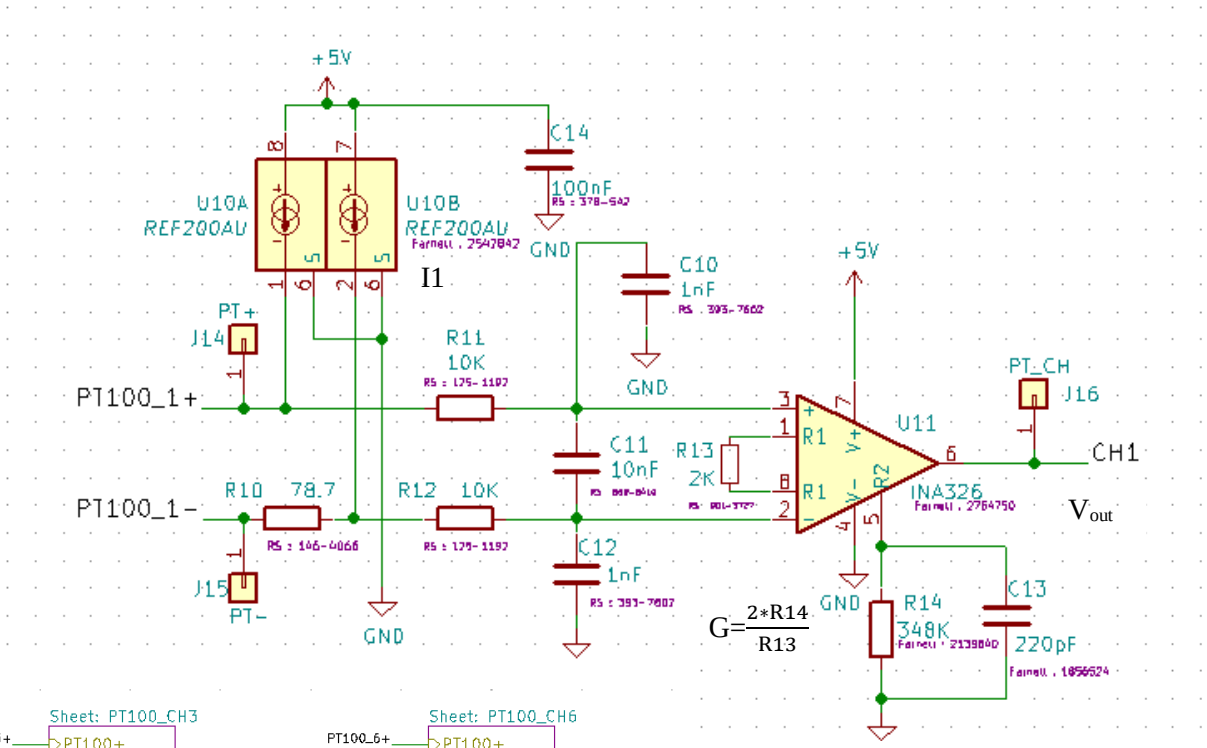
A la suite des tests sur le breadboard, j'ai commencé le schéma de ma solution sur KiCad et fait les feuilles hiérarchiques.

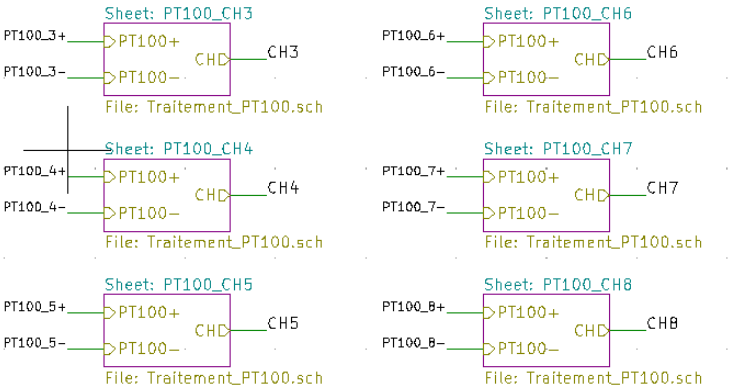
Ces feuilles servent à reproduire plus simplement les montages comme celui des ''entrée PT100'' et ainsi s'il y a besoin de modifications il suffit de modifier le schéma source pour modifier tous les autres.



a. Montage amplificateur

Ci-dessous le montage amplificateur avec les feuilles hiérarchiques



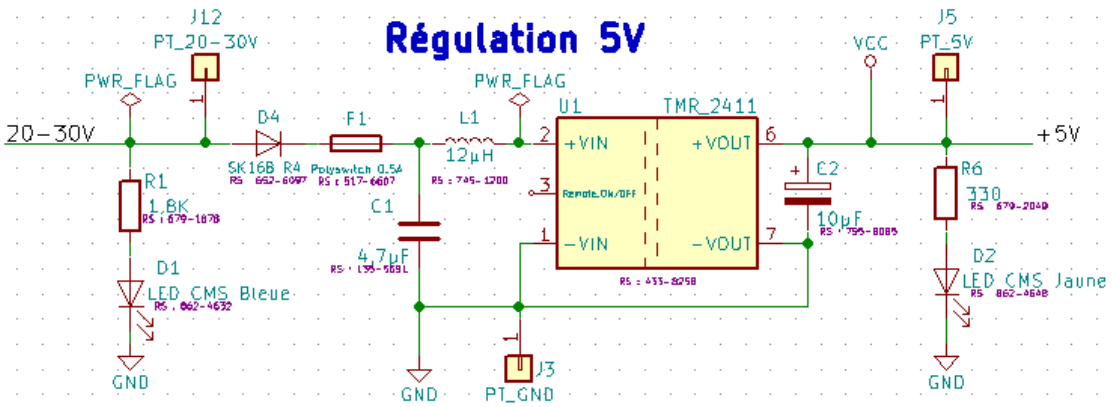


Cette partie permet de relever les températures de la PT100 à partir de la broche 6 de U11.
Les données relevées sont en Volts.
Pour retrouver la résistance de la PT100, j'ai utilisé le calcul suivant :

$$\frac{\frac{Vout}{Gain}+R10*I1}{I1}=Pt100\ (en\ \Omega)$$

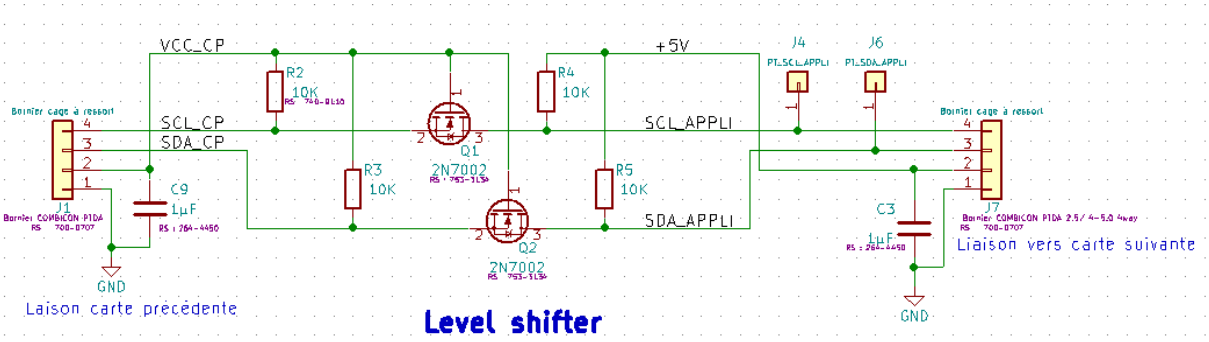
b. Régulation 5V

Ci-dessous la partie Régulation



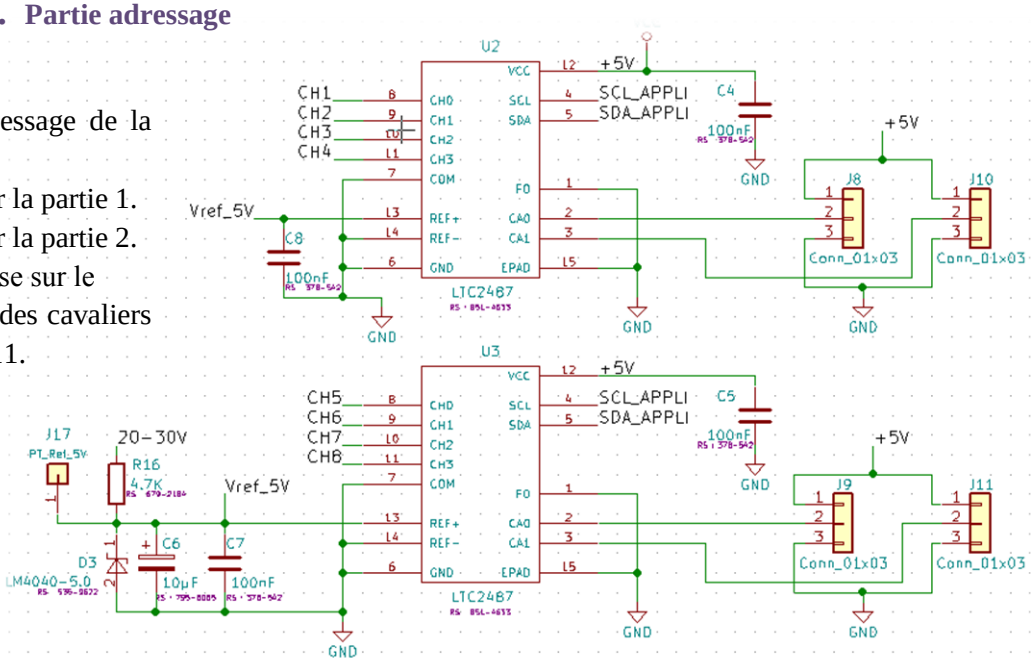
Cette partie nous permet de transformer l'alimentation 24 volts fournis par le module Phoenix en 5 volts pour alimenter les Channels.
Deux Leds sont présentes afin de vérifier la tension d'entré (24V) et la tension de sortie (5V).

c. Level shifter



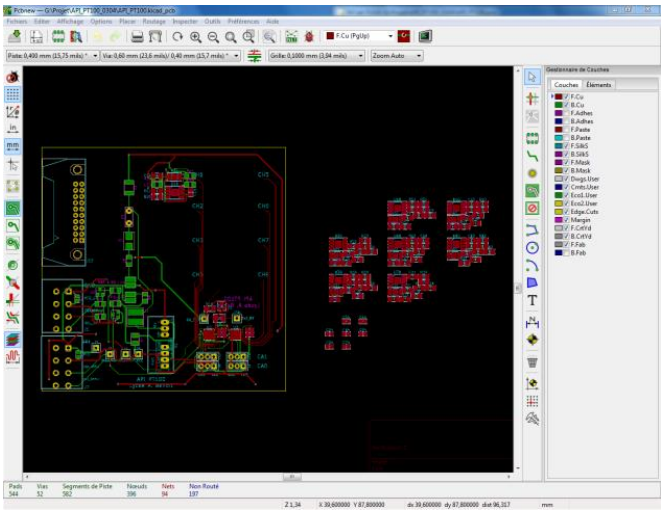
d. Partie adressage

Ci contre la partie adressage de la carte.
Les channels 1-4 sont sur la partie 1.
Les channels 5-8 sont sur la partie 2.
Pour sélectionner l'adresse sur le Logiciel, on doit mettre des cavaliers sur les borniers J10 et J11.

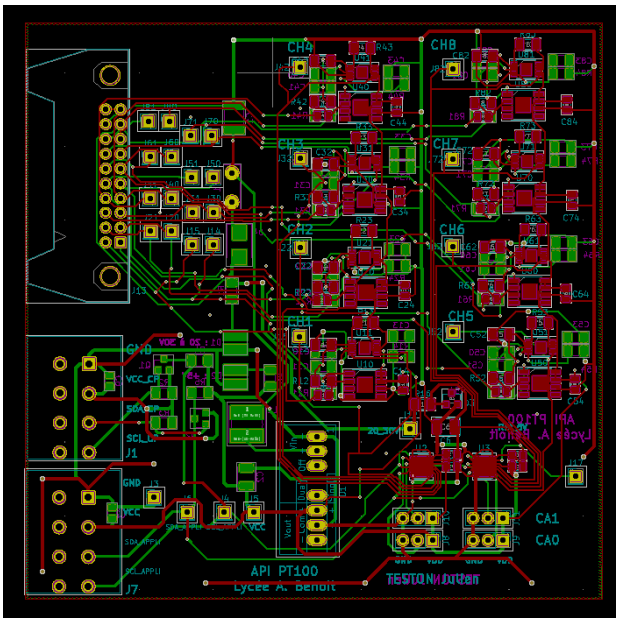


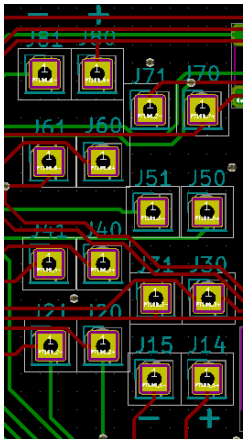
6.9. Routage via KiCad

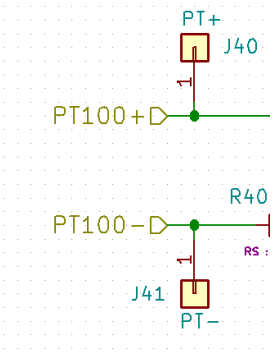
A la suite de la création du schéma, je me suis occupé du routage de la carte. La difficulté, un grand nombre de composants à placer sur une carte de petite dimension imposée par l'entreprise (10x10 cm).



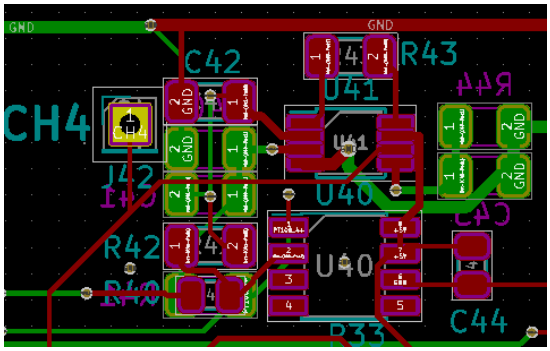
J'ai ajouté des points de test pour chaque channel afin de faciliter la vérification de mesure du 5V, du 24V, des sorties des channels et des sondes PT100.



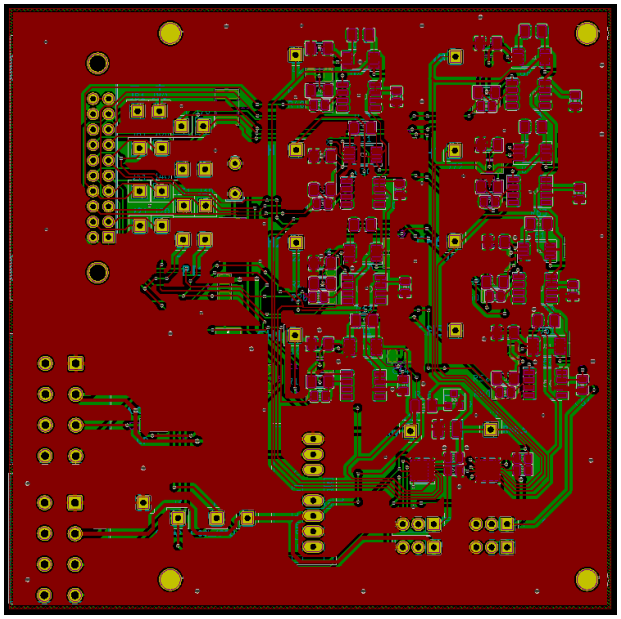




J'ai aussi ajouté des points de tests à la suite du connecteur 2* 10 pour avoir accès au point d'entrée des PT100.
Ainsi l'accès aux valeurs importantes est plus simple.



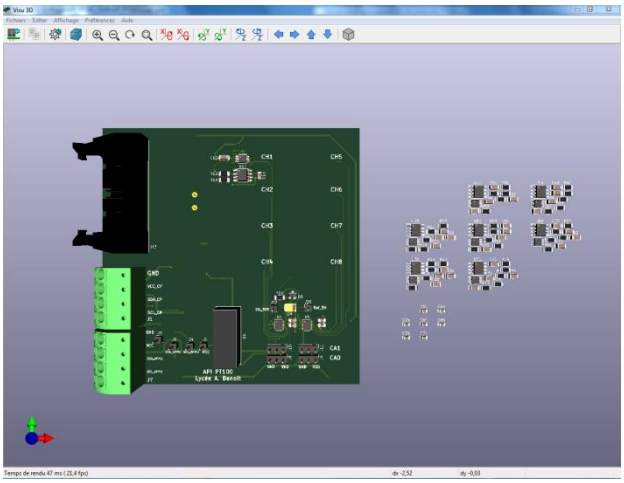
Pour le routage, j'ai dû être extrêmement minutieux car comme mentionné précédemment, la carte est petite pour le nombre de composants à insérer. J'ai donc dû optimiser au maximum l'emplacement des composants.

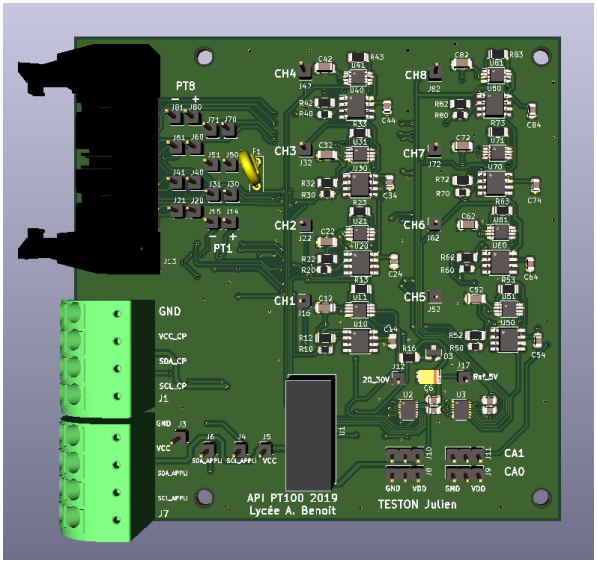


Pour le plan de masse, j'ai modifié l'emplacement de quelques composants afin de couvrir toute la carte.

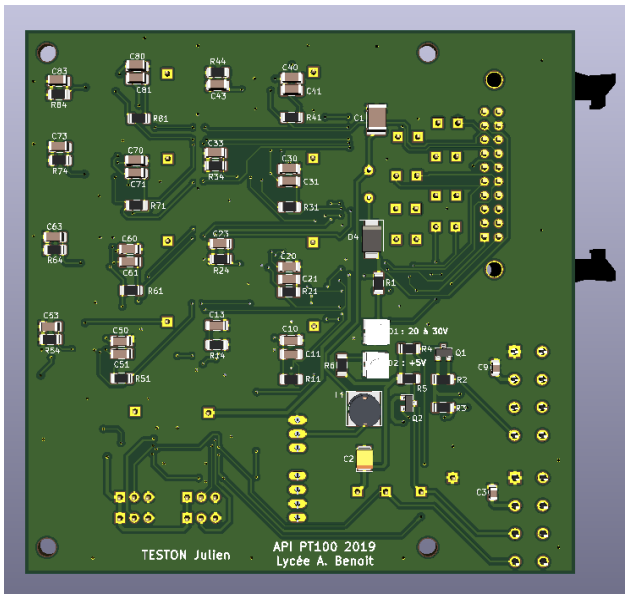
a. Visualisation 3D de la carte

Ci contre on peut voir la carte à moitié routée.
Ci-dessous la carte finie et prête à être commandée.

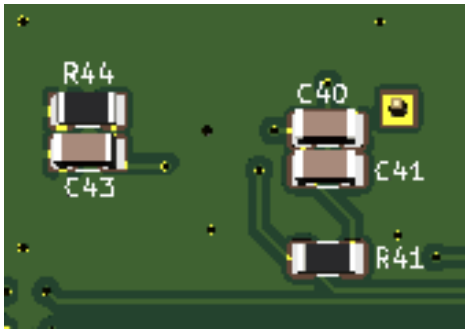


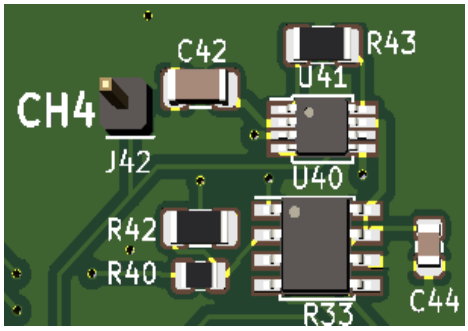


L'entreprise m'a imposé un ordre pour les channels mais pour le reste de la carte, je n'avais pas de directives précises.
J'ai donc placé mes channels sur le coté droit de la carte et fait une masse commune afin de faciliter le routage.
De plus ayant beaucoup de CMS et très peu de « gros composants », j'ai pu en mettre sur les deux face de la carte.



Les points de test étant traversants, ils sont plus simple à intégrer a la carte.





6.10. Soudure de la carte

a. Nomenclature des composants

Le professeur nous a distribué les composants à partir de la liste des composant du schémas que nous lui avons imprimé.
Ici, au total, nous avons 160 composants (122 CSM et 38 traversants)

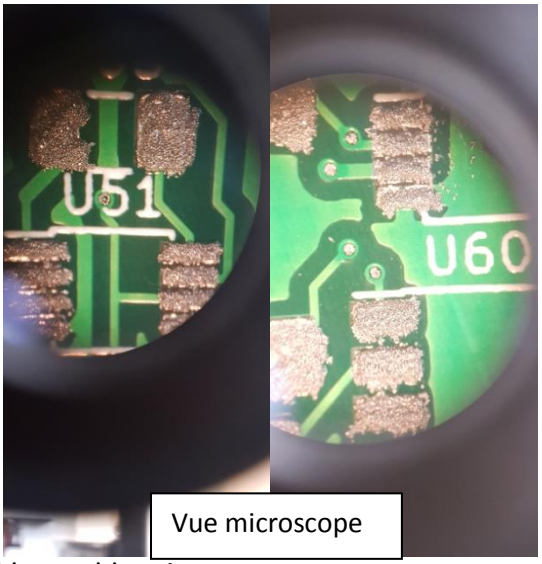
Référence	Valeur	Code com...	Qty
C1	4,7µF	RS : 135-5691	1
> C2, C6	10µF	RS : 795-8085	2
> C3, C9	1µF	RS : 264-4450	2
> C4 - C84	100nF	RS : 378-542	12
> C11 - C81	10nF	RS : 669-8410	8
> C10 - C82	1nF	RS : 393-7602	16
> C13 - C83	220pF	Farnell : 18565	8
D1	LED CMS Bleue	RS : 862-4632	1
D2	LED CMS Jaune	RS : 862-4648	1
D3	LM4040-5.0	RS : 535-9622	1
D4	SK16B R4	RS : 652-6097	1
F1	Polyswitch 0,5A		1
> J1, J7	Bornier cage à ressort	RS : 700-0707	2
J3	PT_GND		1
J4	PT_SCL_APPLI		1
J5	PT_5V		1
J6	PT_SDA_APPLI		1
> J8 - J11	Conn_01x03		4
J12	PT_20-30V		1
J13	Conn_02x10_Odd_Even	RS : 461-118	1
> J14 - J80	PT+		8
> J15 - J81	PT-		8
> J16 - J82	PT_CH		8
J17	PT_Ref_5V		1
L1	12µH	RS : 745-1200	1
> Q1, Q2	2N7002	RS : 753-3134	2
R1	1,8K	RS : 679-1878	1
> R2, R3	10K	RS : 740-9110	2
> R4, R5	10K	RS :	2
R6	330	RS : 679-2049	1
> R10 - R80	78.7	RS : 146-4066	8
> R11 - R82	10K	RS : 125-1192	16
> R13 - R83	2K	RS : 901-3727	8
> R14 - R84	348K	Farnell : 21396	8
R16	4.7K	RS : 679-2184	1
U1	TMR_2411	RS : 433-8258	1
> U2, U3	LTC2487	RS : 851-4633	2
> U10 - U80	REF200AU	< ... >	8
> U11 - U81	INA326	Farnell : 27647	8
> U20 - U80	REF200AU	Farnell : 25428	8

b. Comment souder les composants

Pour souder, nous utilisons une pâte à braser.
Pour appliquer la pâte sur la carte, j'ai utilisé un stencil qui est fourni lors de la commande de la carte.
Il faut, avant toutes choses, bien caler la carte afin que dès que l'on pose le stencil dessus, les empreintes de la carte et celle du stencil correspondent bien.

Ensuite, nous passons la pâte à braser sur le stencil à l'aide d'une spatule afin que la pâte se dépose sur les empreintes de la carte. Puis il faut vérifier s'il y a assez de pâte sur les empreintes.
Enfin, il faut juste poser les CMS sur leurs empreintes respectives.

J'ai eu un problème avec deux pistes lors des soudures des INA car leurs empreintes étant très fines, la pâte à braser les a noyé (voir U51) ; j'ai dû séparer les empreintes de pâte à l'aide d'une lame de cutter un travail très pointilleux sous microscope (U60 après séparation).

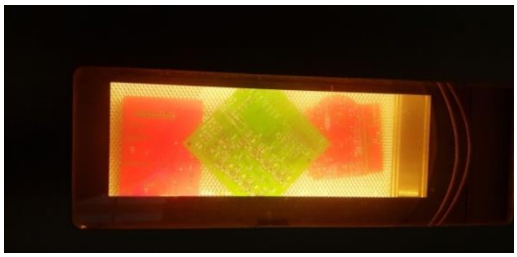


Vue microscope

Une fois les CMS installés, j'ai contrôlé au microscope qu'ils soient bien positionnés.

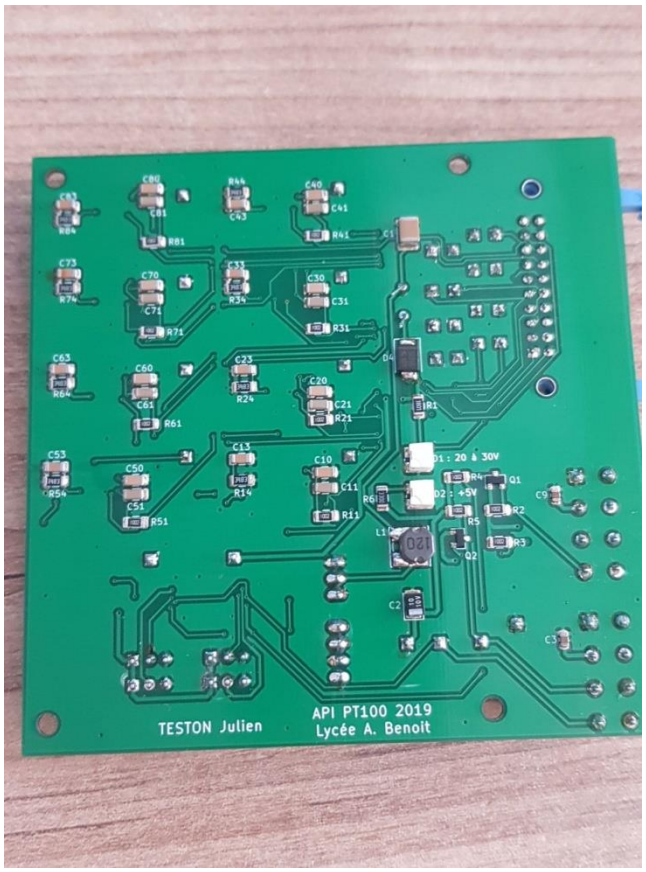
Puis j'ai mis la carte dans le four et lancé le programme 2

- ✓ Préchauffage 90 °C
- ✓ 2 minutes 30 secondes montée à 220 °C
- ✓ 1 minute montée à 250 °C (refusions)
- ✓ 1 minute 50 secondes : refroidissement



Une fois la carte refroidie, j'ai regardé au microscope s'il n'y avait pas de soucis au niveau des soudures.
Pour souder le deuxième coté de la carte il faut la surélever à l'aide de vielle cartes pour éviter que les CMS déjà soudés soient en contact avec la grille.
Pour finir, on soude les composants traversants à l'aide d'un fer à souder et d'étain après avoir vérifié les soudures de la seconde face.

c. Première soudure CMS



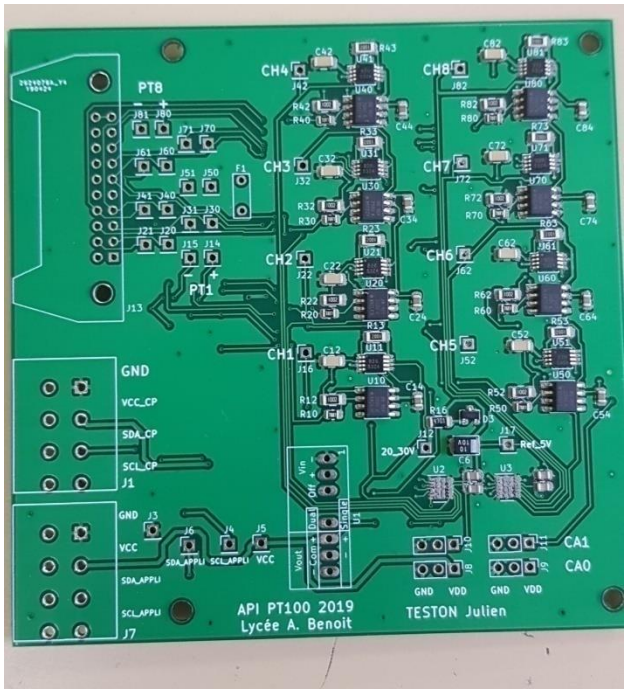
J'ai commencé par le verso de la carte pour éviter de passer les INA et REF deux fois au four.

Cette face comporte :

- Les résistances R11 à R81, R14 à R84 ainsi que R1, R2, R3, R4 et R6
- Les condensateurs C10 à C80, C11 à C81, C13 à C83 ainsi que C1, C2, C3 et C9
- Les transistors Q1 et Q2
- Les Leds D1 et D2
- La Diode D4

  La bobine ne pouvant pas passer au four, elle sera soudée à l'aide d'un fer à souder à la fin.

d. Seconde soudure CMS



La face avant contient :

- Les résistances R10 à R80, R12 à R82, R13 à R83 ainsi que R16
- Les condensateurs C12 à C82 et C14 à C84 ainsi que C4, C5, C6, C7 et C8
- LM4040 D3
- Les INA326 U11 à U81
- Les REF200 U10 à U80
- Les convertisseurs LTC2487 U2 et U3

e. Soudure des traversants

Pour finaliser la carte, il me restait juste à souder les traversants :

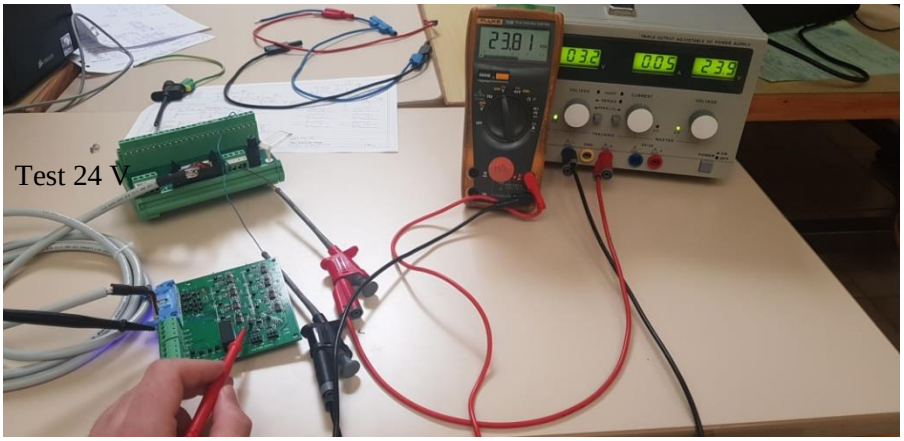
- Point de test :
  - Masse J3
  - SCL J4
  - 5V J5
  - SDA J6
  - 24V J12
  - Ref 5V J17
  - Polyswitch F1
  - PT- J21 à J81 ainsi que J15
  - PT+ J20 à J80 ainsi que J14
  - PT Channel J22 à J82 ainsi que J16

- Convertisseur DC/DC U1
- Connecteur 02*10 J13
- Connecteur 01*03 J8, J9, J10, J11
- Borniers cage à ressort J1 et J7
- La bobine qui n'est pas un traversant mais qui ne passe pas au four doit être soudée à la main.

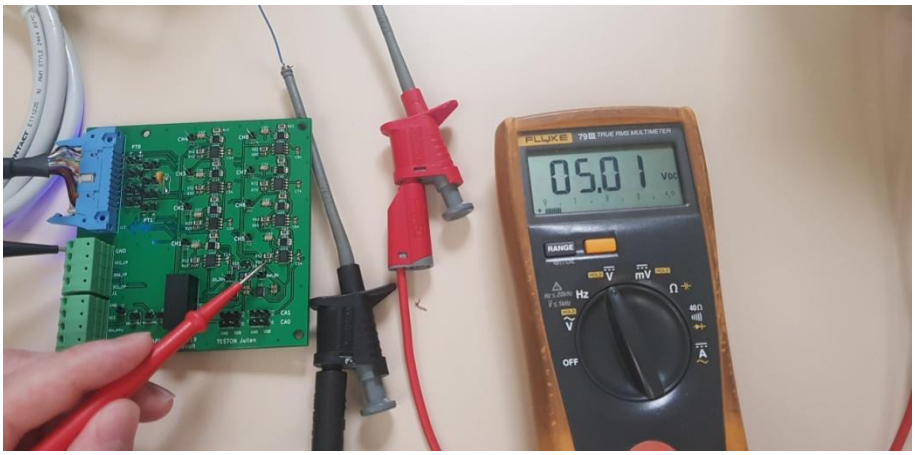
f. Tests finaux

Après avoir fini de souder la carte, j'ai vérifié la continuité des soudures et mis la carte sous tension par le module Phoenix. A l'aide d'un multimètre, j'ai vérifié les tensions grâce au point de test (24V, réf 5V, 5V et les points de test des Channels)

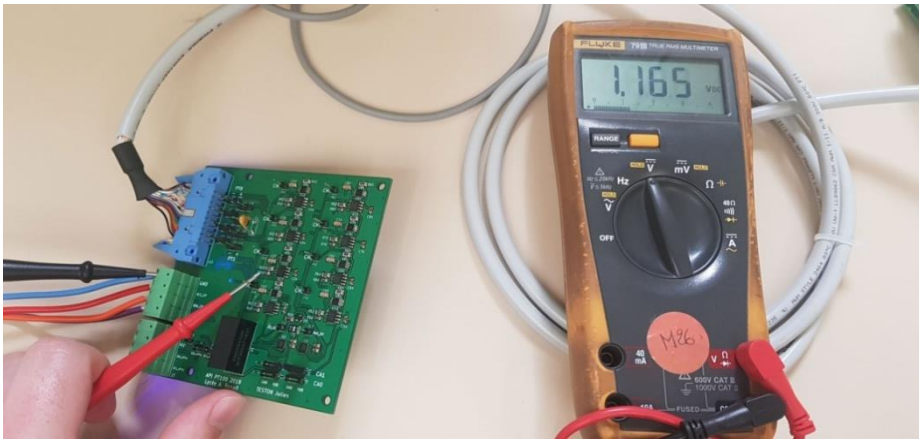
Test 24 V



Test 5V



Une fois les tensions validées, j'ai branchée ma PT100 sur le channel 1 et j'ai relevé la tension de sortie du Channel 1 qui était satisfaisante.



6.11. Expérimentation de la carte API

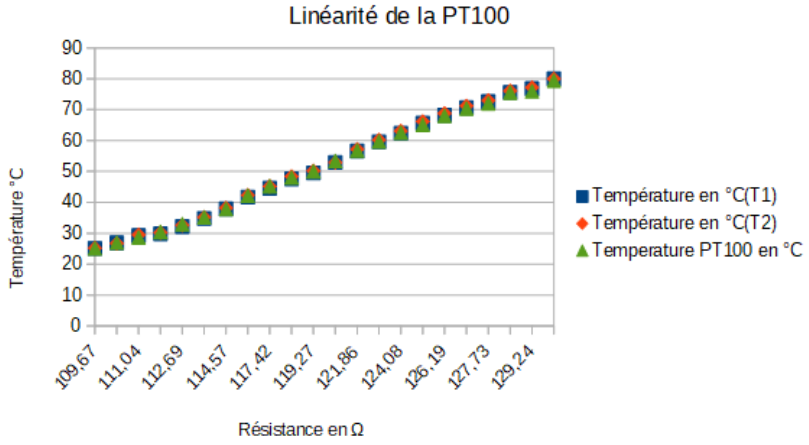
Afin de tester la carte API, et avant de finaliser la carte, nous avons créé une situation dans laquelle la température variait de manière linéaire

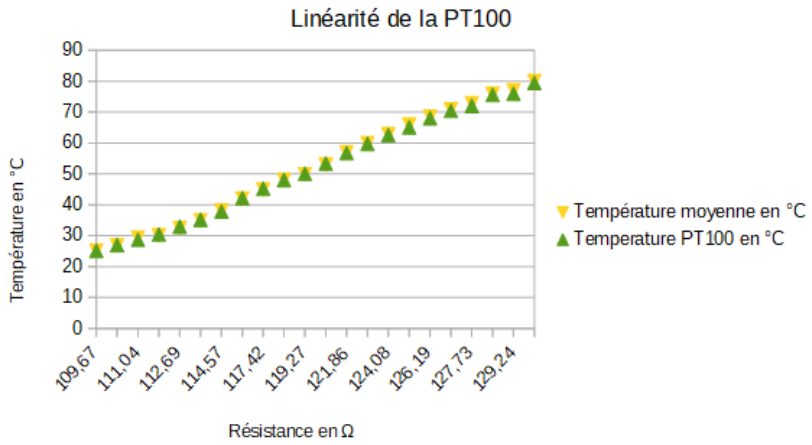
Matériel utilisé : Chauffe eau, 2 thermomètres, Carte API de mise en situation (seulement 2 channels soudés).

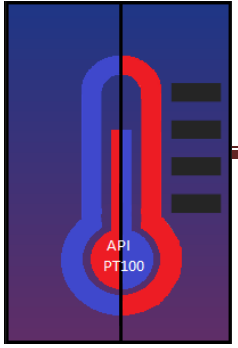
La carte n'ayant pas le convertisseur ni les autres composants, je l'ai alimenté par une alimentation 5V.

Pour cette mise en situation, j'ai immergé la PT100 et les deux thermomètres dans de l'eau à température ambiante puis j'ai fait chauffer l'eau jusqu'à 80°C de manière linéaire pour voir si le montage suivait précisément les thermomètres.

Résistance PT100 en Ω	Température en °C(T1)	Température en °C(T2)	Température moyenne en °C	Temperature PT100 en °C	Tension relever en Vout
109,67	25,1	25,2	25,15	25,11	1,087
110,38	26,9	26,7	26,8	26,96	1,112
111,04	29,2	29,5	29,35	28,67	1,135
111,72	29,8	30,2	30	30,44	1,159
112,69	32,2	32,6	32,4	32,96	1,193
113,54	34,8	35,2	35	35,18	1,223
114,57	37,9	38,2	38,05	37,84	1,259
116,25	41,7	42,3	42	42,21	1,318
117,42	44,6	45,2	44,9	45,24	1,359
118,50	47,6	48,4	48	48,05	1,397
119,27	49,5	50,2	49,85	50,05	1,424
120,55	52,9	52,9	52,9	53,38	1,469
121,86	56,6	57,2	56,9	56,79	1,515
123,00	59,6	60,2	59,9	59,75	1,555
124,08	62,4	63,1	62,75	62,56	1,593
125,05	65,6	66,2	65,9	65,07	1,627
126,19	68,2	68,8	68,5	68,03	1,667
127,10	70,6	71,2	70,9	70,40	1,699
127,73	72,6	73	72,8	72,03	1,721
129,10	75,6	76,1	75,85	75,58	1,769
129,24	76,8	77,2	77	75,95	1,774
130,58	80	80	80	79,43	1,821







Pour remplir le tableau de données je me suis servis de la tension relevé V_{out} à l'aide du calcul utilisé précédemment j'ai tout d'abord trouvé la valeur en ohms de la PT100

Rappel $\frac{\frac{Vout}{Gain}+R10*I1}{I1}$

Soit pour la mise en situation :

$$\frac{\frac{1.087}{351}+78.7*100.10-6}{100.10-6}=109.67\Omega$$

Pour avoir la température de la PT100 j'ai utilisé la formule suivante :

$R\theta=100(1+\alpha\theta)$

Avec :

- θ = Température en ℃
- α = 0.00385 coefficient de température (Ω / ℃) de la sonde Standard Européen.
- $R\theta$ = Résistance de la PT100 en Ω

Etant donné que la valeur cherché est la température j'ai du changer mon calcul de façon a obtenir celui-ci :

$$\frac{\frac{R\theta}{100}-1}{\alpha}=\theta$$

Dans la mise en situation nous avons donc :

$$\frac{\frac{109.67}{100}-1}{0.00385}=25.11\ ℃$$

Pour la température moyenne j'ai simplement fais la moyenne des températures relevées sur les deux thermomètres.

Cette mise en situation m'a permis de valider le cahier des charges de la société en montrant que la solution choisie fonctionnait. J'ai alors pu finir de souder ma carte définitive à 8 channels.
Le calcul utilisé dans le programme est le suivant :
((((Vout/G)+R10*I1)/I1)/100-1)/α = θ
Soit :
((((1.087/349)+78.7*100.10^{-6})/100.10^{-6}/100-1)/0.00385 = 25.11

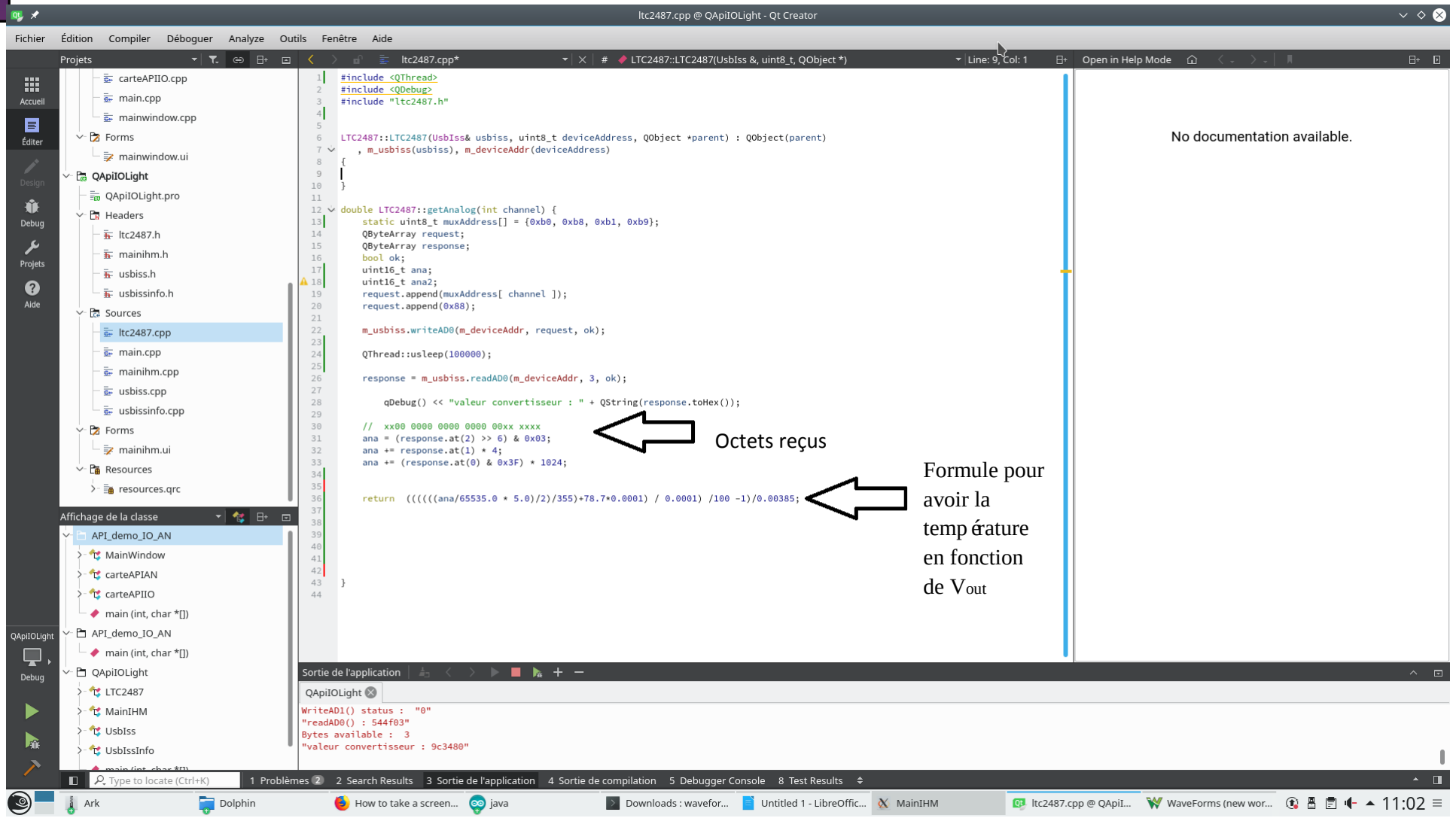
6.12. Finalité du projet

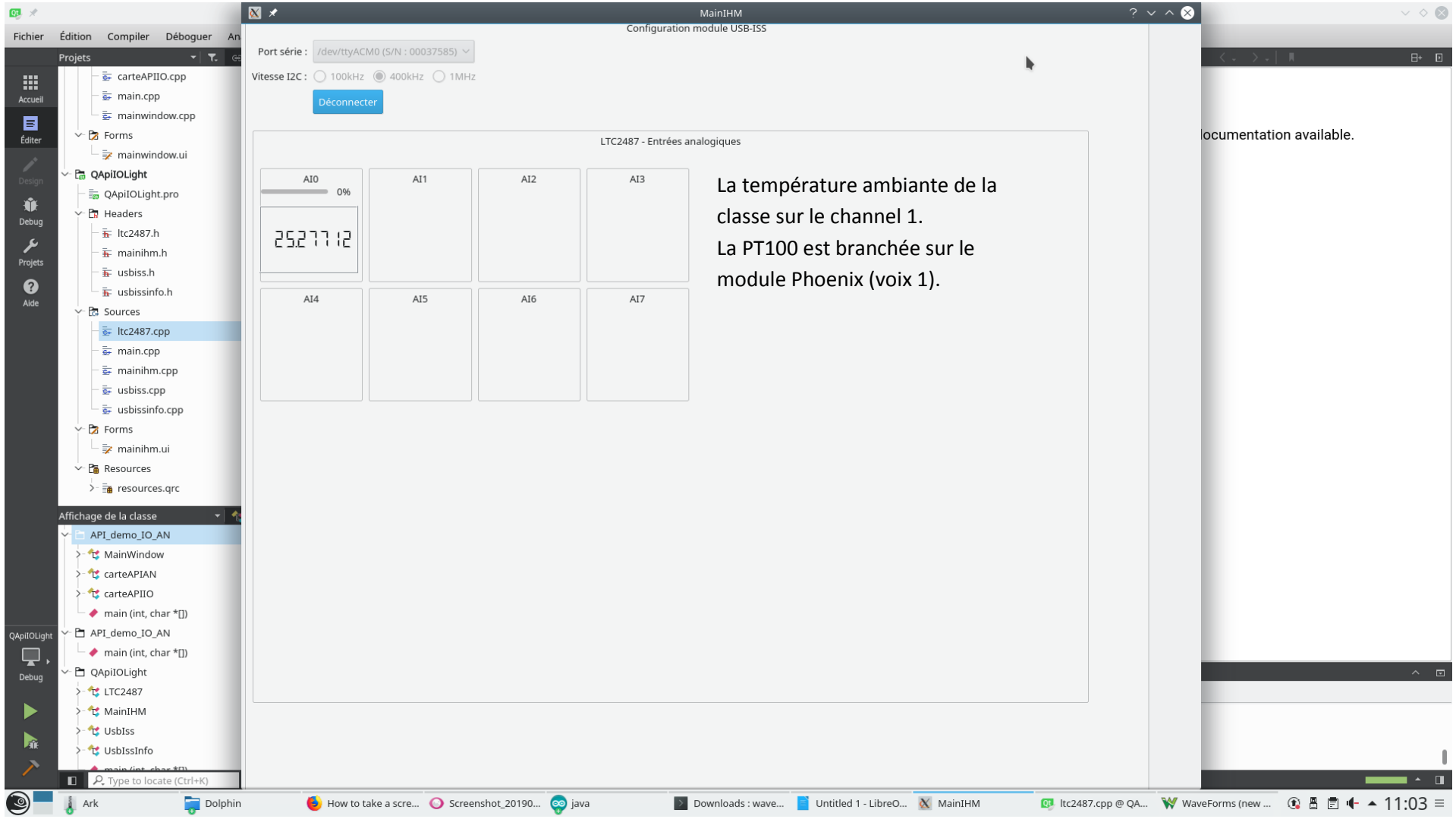
L'entreprise nous a demandé :

- ✓ Une carte de 10cm*10cm
- ✓ 8 channels
- ✓ Le budget de 350€ est respecté
- ✓ La précision du montage est suffisante a l'utilisation demandée
- ✓ La prise de mesure sur la carte est simple
- ✓ Le module Phoenix est utilisé
- ✓ Les applications fonctionnent sous Linux ainsi que Raspberry Pi
- ✓ Les PT100 peuvent être utilisées à des distances différentes
- ✓ Les cartes peuvent être chainées entre elles et l'adressage I2C se fait simplement
- ✓ L'alimentation est faite par le module Phoenix
- ✓ La Carte API TESTON fonctionne les tests effectués sont satisfaisant

- ✓ Le cahier des charges est donc respecté

6.13. Application Qt




MainIHM
Configuration module USB-ISS
Port série :
/dev/ttyACM0 (S/N : 00037585)
Vitesse I2C :
100kHz
400kHz
1MHz
Déconnecter
LTC2487 - Entrées analogiques
AI0
0%
25.277712
AI1
AI2
AI3
AI4
AI5
AI6
AI7
La température ambiante de la classe sur le channel 1.
La PT100 est branchée sur le module Phoenix (voix 1).
Projets
carteAPIIO.cpp
main.cpp
mainwindow.cpp
Forms
mainwindow.ui
QApiIOLight
QApiIOLight.pro
Headers
ltc2487.h
mainihm.h
usbiss.h
usbissinfo.h
Sources
ltc2487.cpp
main.cpp
mainihm.cpp
usbiss.cpp
usbissinfo.cpp
Forms
mainihm.ui
Resources
resources.qrc
Affichage de la classe
API_demo_IO_AN
MainWindow
carteAPIAN
carteAPIIO
main (int, char *[])
API_demo_IO_AN
main (int, char *[])
QApiIOLight
LTC2487
MainIHM
UsbIss
UsbIssInfo
Type to locate (Ctrl+K)
Ark
Dolphin
How to take a scre...
Screenshot_20190...
java
Downloads : wave...
Untitled 1 - LibreO...
MainIHM
ltc2487.cpp @ QA...
WaveForms (new ...
11:03

7. CONCLUSION

Ce projet m'a permis d'améliorer mes compétences dans la création de carte électroniques, et dans la lecture de schéma structurel mais aussi de consolider mes connaissances en programmation, de plus le travail en équipe ainsi que la rédaction de documents communs m'on permit d'augmenter mon esprit d'équipe.

En ce qui concerne la suite du projet, il faut que je finisse la mise en œuvre du programme qui va permettre l'affichage direct des températures. Pour l'instant, le programme ne permet que de réceptionner la sortie des channels soit une donnée en volts.

Je vais pour cela utiliser les calculs permettant de transformer les volts en ohms puis les ohms en degrés Celsius.

EC1 BURLE Julien

7.1. Présentation du projet EC1

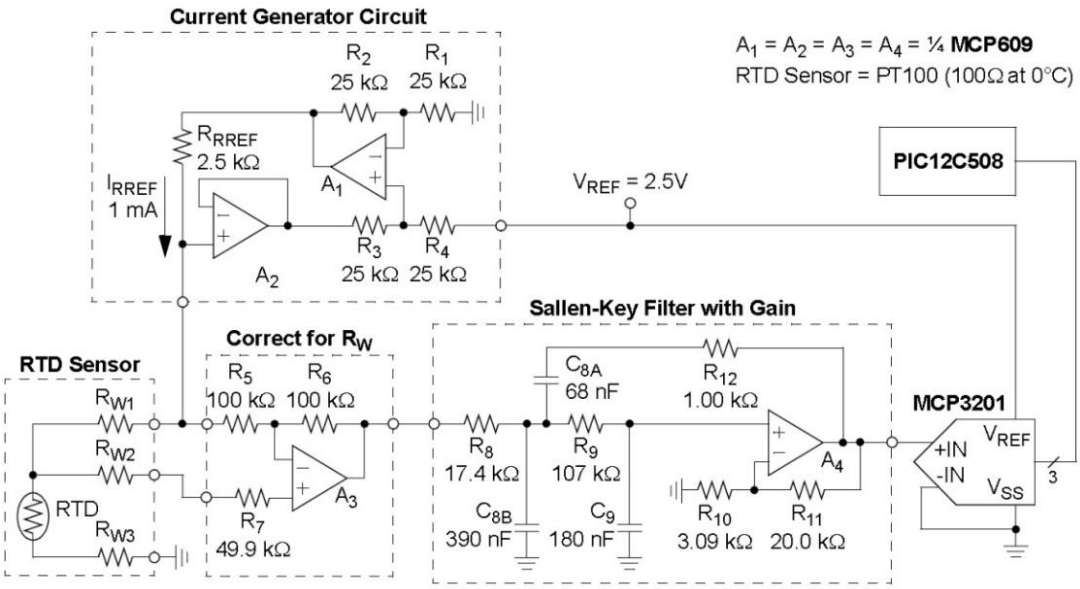
Ce projet comporte le travail de trois étudiants, dont 2 EC et 1 IR

Notre première tâche était d'analyser le travail des étudiants de l'année 2018, et de prendre donc compte des différents dossiers, schémas, programmes, solutions et autres informations

Le projet comporte deux solutions, la solution Microchip et la solution Ti

J'ai été assigné à la solution Microchip, et Teston Julien, l'étudiant EC2, a été assigné à la solution Ti

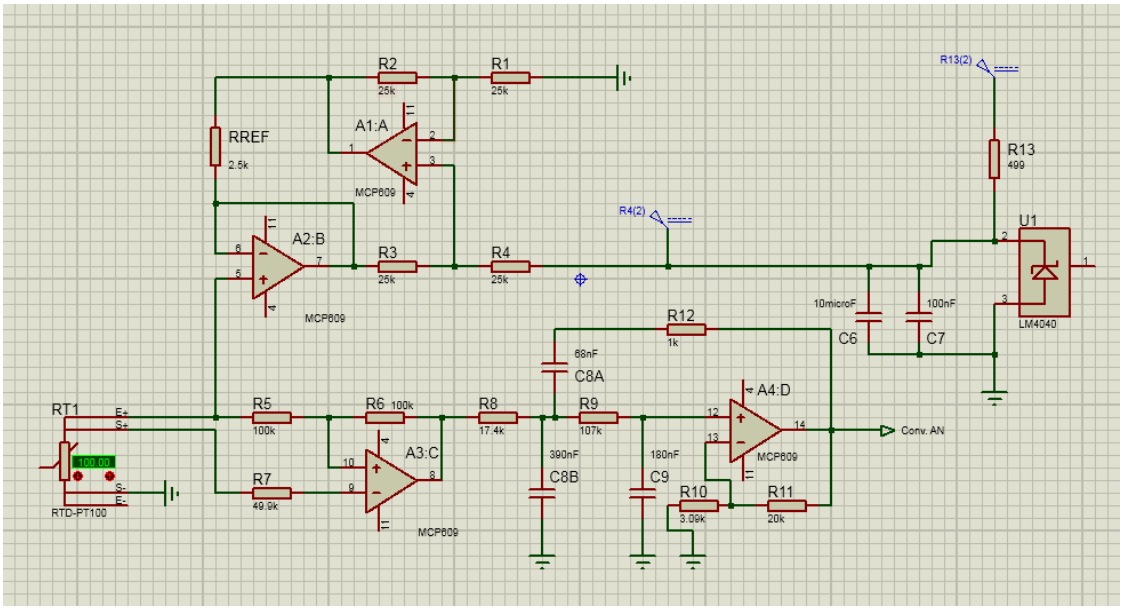
Nous aurons donc à choisir une des solutions plus tard dans le projet, et nous expliquerons nos choix



Ce schéma illustre la solution Microchip, mais le résultat final sera différent (voir le schéma sur Proteus 8), on remplace notamment le MCP3201 et le PIC12C508 par un LM4040, qui sera relié à la masse, avec deux condensateurs, ainsi qu'à la tension de référence (VREF) de 2.5V, et l'alimentation de 5V, avec une résistance de 500Ω (R13). L'amplificateur A4 sera donc relié au convertisseur Analogique Numérique.

Calcul de R13 = (5-2.5)/0.005 = 500Ω

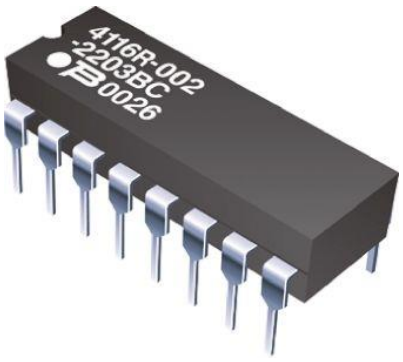
7.2. Schéma Proteus



7.3. Les choix de matériels

L'amplificateur A4 fournissant un gain, et le convertisseur Analogique Numérique ayant une plage de tension entre 0.138V et 2.343V pour une tension référence de 5V, on pourrait avoir besoin de changer les deux résistances R10 et R11 pour bien avoir une tension supérieure à 0V et inférieure à 2.5V pour le convertisseur.

On pourrait aussi changer les résistances R5, R6 et R7 ; les deux résistances R5 et R6 doivent avoir la même valeur, car la valeur de la résistance R7 sera toujours deux fois plus petites que R5 ou R6. Le rôle de ces résistances est d'équilibrer les intensités I- et I+ d'A3, on pourrait donc changer les valeurs de R5 et R6 pour 50kΩ, la valeur de R7 serait donc de 25kΩ



Les quatres résistances R1 à R4 pourraient aussi être remplacées par un réseau de résistance ;

Si une des résistances chauffait plus que les trois autres, cela pourrait poser un problème dans les calculs, un réseau de résistances résoudrait ce problème, car les résistances chaufferaient autant les unes que les autres, on aurait donc des valeurs équivalentes.

7.4. Avantages et inconvénients par rapport à la solution Ti

L'attrait de la solution Microchip est son moindre coût ;

La solution Ti utilise des composants onéreux, comme le REF200 et l'INA326, avec ~8€ HT par pièce pour le REF200, et entre 4 et 5€ HT par pièce pour l'INA326, multipliés par 8, ces deux seuls composants nous donnent une facture d'environ 100€

Tandis que la solution Microchip utilise des composants comme le MCP609, à un peu plus d'1€ HT par pièce, et il n'y a besoin que d'un composant LM4040 pour les 8 sondes PT100 prévues

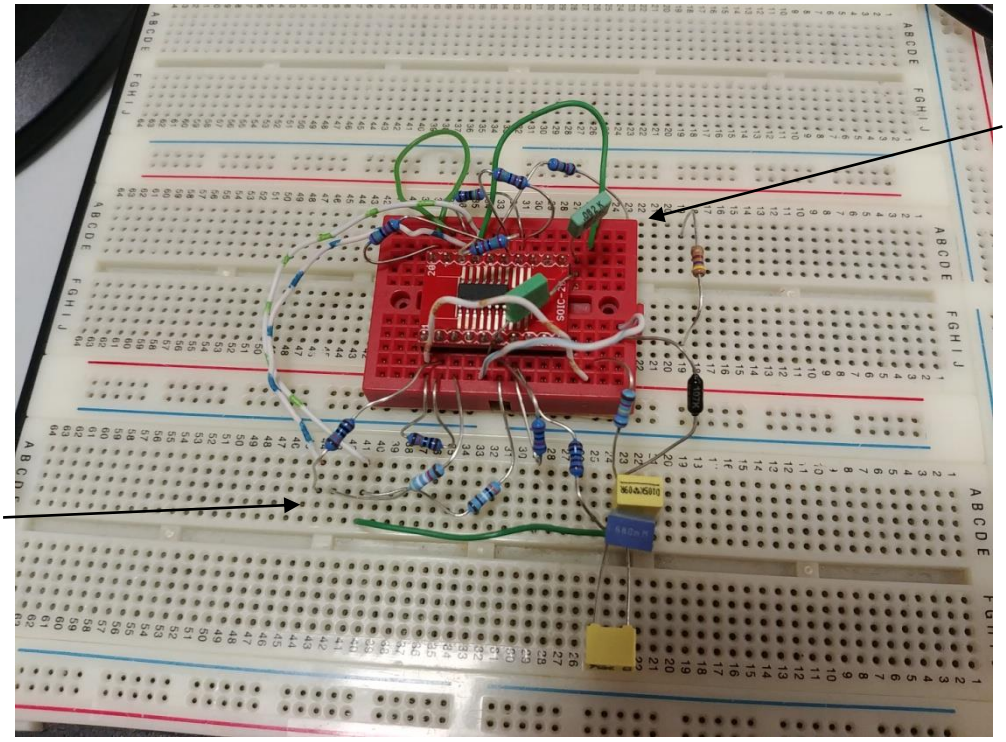
La solution Microchip crée quand même quelques problèmes ;

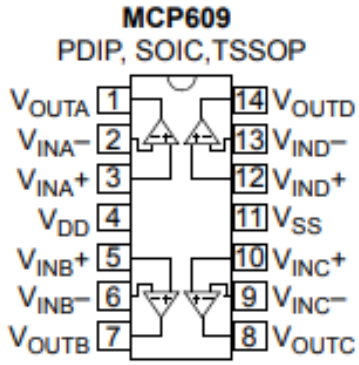
Si utilisée avec un module Phoenix à 2 fils pour chaque sonde et un point commun, l'intensité I REF risque d'être multipliée par le nombre de PT100 sur le circuit, et cela créerait une différence d'environ 0.4 °par sondes additionnelles, ceci est dû au point commun utilisé du module Phoenix

Différentes solutions pourraient être choisies, comme trouver un module Phoenix avec 3 fils pour chaque sonde, ou compenser logiciellement le défaut.

Il y a aussi le problème des longueurs de câbles des sondes quand on utilise la solution Microchip avec le module Phoenix, contrairement à la solution Ti avec le REF200.

7.5. Câblage de la solution Microchip





On peut voir ici que l'amplificateur CMOS MCP609 regroupe les 8 amplificateurs qui sont utilisés dans la solution Microchip

7.6. Erreurs de câblage

Lors du test du câblage rapide, le professeur et moi avons constatés une différence non-négligeable, le capteur nous donnant une valeur d'environ 51,5°, ou 120 ohms, alors que la température ambiante est de 24 °, ou 109,35 ohms

Nous suspectons l'ampliope de ne pas être très précis, mais nous avons aussi trouvés que la longueur du fil connectant la masse du circuit à la masse de l'alimentation causait une partie de la différence trouvée au début du test, passant de 120 ohms (ou 120mV) à 115 ohms/mV, mais la différence est toujours présente

Nous avons ensuite constaté que les résistances de 25Kohms et 2,5Kohms, étaient en fait des résistances de 24Kohms et 2,4Kohms, et que je n'avais pas rajouté de résistances en série pour corriger l'erreur

Et comme espéré, l'ajout de résistances en séries pour arriver à 25Kohms et 2,5Kohms a encore diminué l'erreur de 4mV (120mV au début ; 11mV d'erreur, 115mV après avoir changé le fil de la masse ; 6mV d'erreur, et maintenant 112mV, après avoir rajouté des résistances en séries ; entre 2 et 3mV d'erreur)

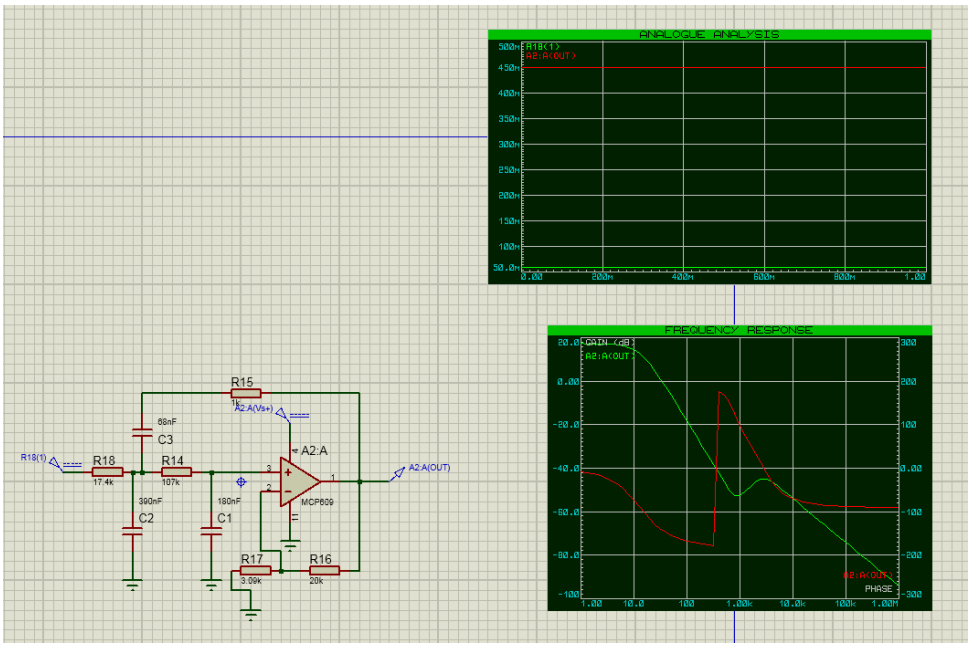
Le reste est sûrement du au câblage en l'air, surtout à cause des longueurs de fils, et avec ces corrections, on arrive à une marge d'erreur de 1 ohm, ou 1mV, donc ~3°C d'erreur

7.7. Raisons de l'abandon de la solution Microchip

Tout d'abord, il y a le problème de compatibilité avec le module Phoenix, qu'API préfèrerait utiliser

Il y a aussi le fait que le schéma de routage de la solution Microchip aurait été un problème, avec une quinzaine de composants pour chaque sonde, on ne pourrait pas ajouter 8 sondes sur la carte

Puis, la plage de tension en sortie convertisseur analogique est non satisfaisante, pour une plage de -100 à 200°C, la tension minimale est de 450mV (voir l'image ci-dessous) et maximale de 1,32V, avec une différence de 0,87V

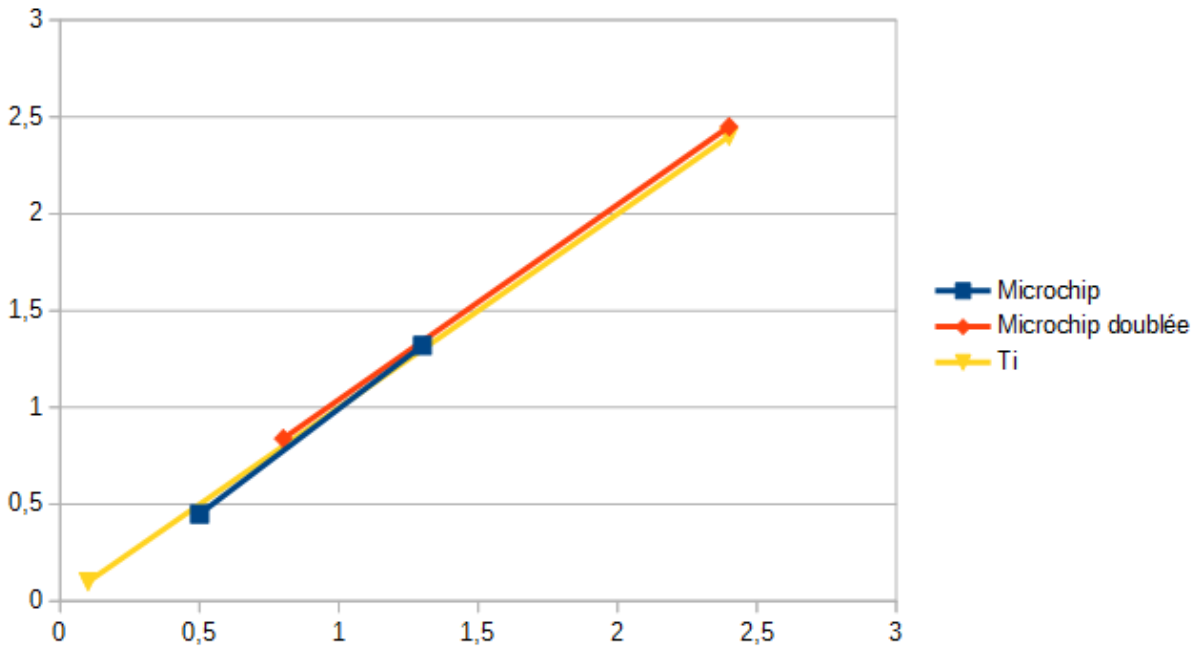


Même si on double l'amplification (en changeant la valeur de R17 à 1.55k par exemple) on arrive a 840mV et 2,45V, donc une plus grosse différence de 1,61V, mais toujours non satisfaisante, on regardera alors la solution TI, qui commence à ~0,1V

7.8. Précision des deux solutions

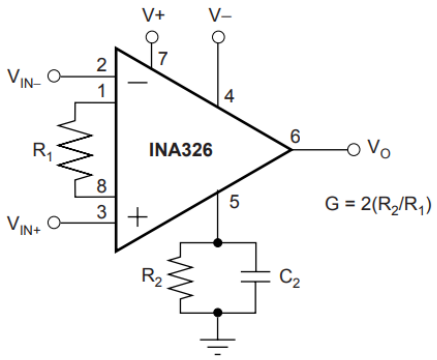
La précision s'élève avec une plage de température de -50 à 150 °C, donc une plage de tension entre 600mV et 1,2V, à 3mV par degrés (différence de 0,6V / par 200 °) pour la solution Microchip

Alors que la solution Ti utilise une meilleure plage de tension, ~0,1V à ~2,4V, donc une différence de 2,3V, et une précision d'environ 12mV par degrés (2,3V / 200°), donc une précision 4 fois supérieure, et même si plus chère, cette solution pourra voir son prix baisser avec des commandes Aliexpress



On peut donc voir ici que la solution Ti occupe toute la plage de tension, et est donc plus précise, alors que la solution Microchip, même doublée, n'arrive pas à occuper entièrement la plage de tension

7.9. Composants Ti

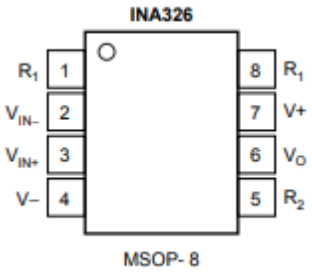


L'INA326 est un amplificateur d'instrumentation de haute performance peu cher, avec broches d'entrées et de sortie

Ce sont des amplificateurs à une seule alimentation et ont des marges d'erreurs faibles en courant continu, ils peuvent être utilisés pour des applications générales à des applications nécessitant de la précision

Le composant a une plage de tension allant de 2.7V à 5.5V, et une plage de température de -40 °C à 125 °C

Le gain peut être modifié en changeant les résistances R1 et R2 indiqués sur l'image

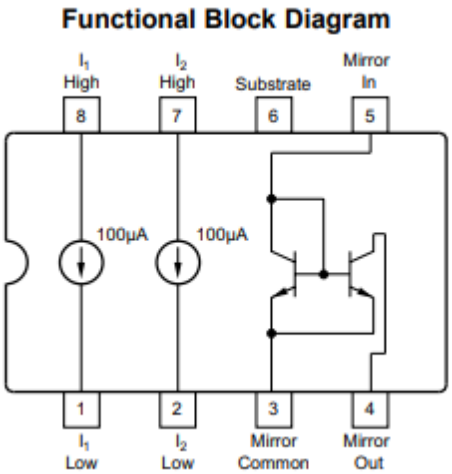


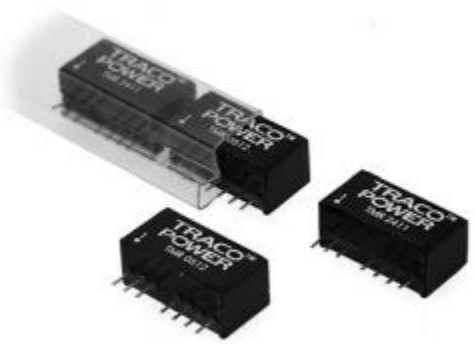
Le REF200 est un régulateur de courant

Sa plage de tension est grande, allant de 2.5V à 40V, et a une plage de tension similaire au INA326, allant de -40 °C à 85 °C

Le REF200 nous permettra de créer deux sources de courant précises de 100 µA

Ces sections sont isolées, et sont donc complètement indépendantes, elles peuvent aussi être réglées pour produire des références de courant de 50 µA, 200 µA, 300 µA ou 400 µA





La série TMR 2 est une famille de convertisseur DC/DC de 2W avec sorties régulées

Sa plage de température est aussi similaire aux deux précédents composants, allant de -40 °C à 85 °C

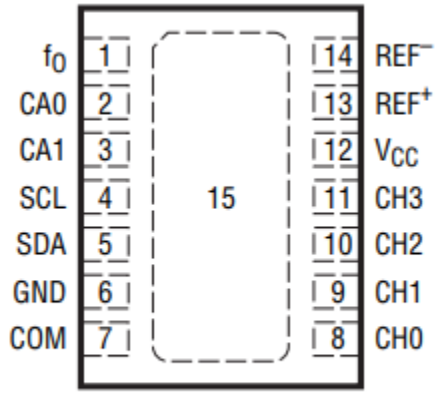
Le TMR 2411 est un convertisseur DC/DC avec une tension d'alimentation nominale de 24V DC

Il régulera la tension d'alimentation de 24V et nous permettra d'avoir un VCC égal à 5V

Le LTC 2487 est un convertisseur analogique numérique à quatre channels de résolution 16 bits possédant une interface i2c

Son schéma d'échantillonnage élimine les erreurs de courant d'entrée dynamiques

Le LTC 2487 inclus un gain programmable, une sonde de température à haute précision et un oscillateur intégré, la sonde de température a résolution de 1 à 2°C et une marge d'erreur absolue de 2°C

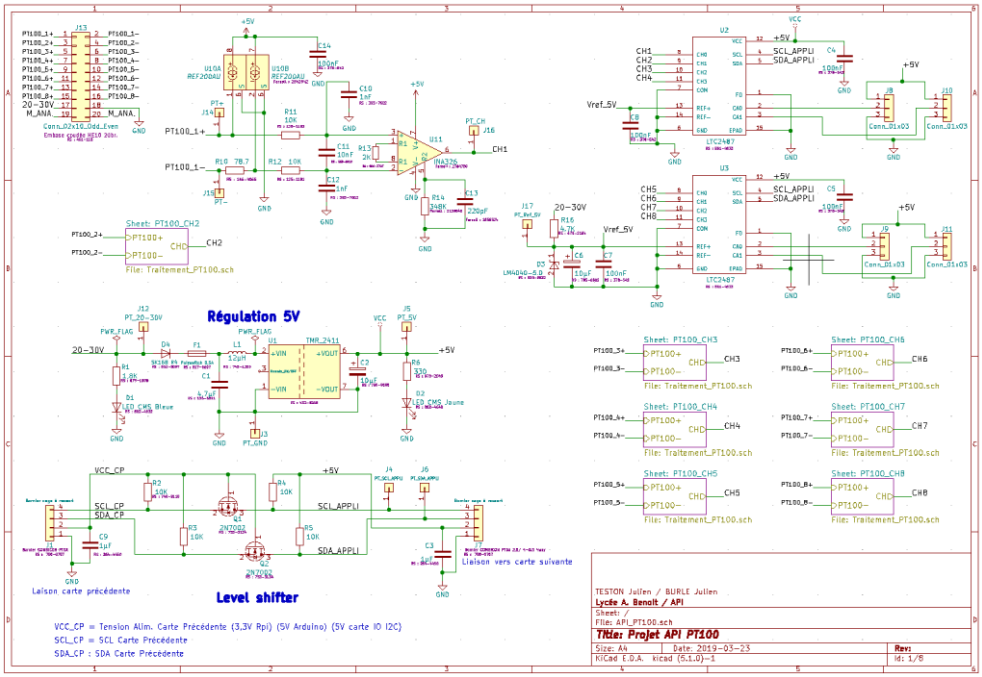


DE PACKAGE
14-LEAD (4mm × 3mm) PLASTIC DFN

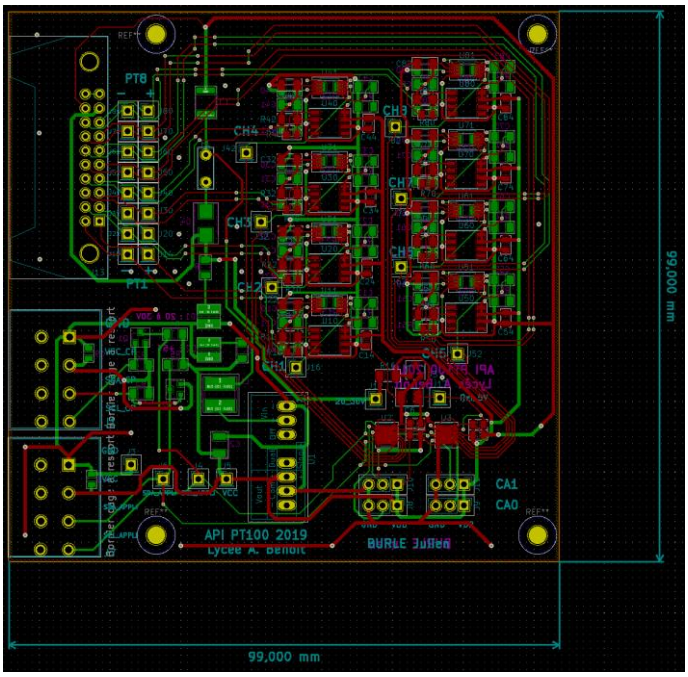
7.10. Schéma et routage Kicad

Après avoir repris la solution Ti, il fallait reprendre les dossiers des anciennes cartes AN API et appliquer notre solution au schéma et changer le routage

Schéma structurel :



Routage :

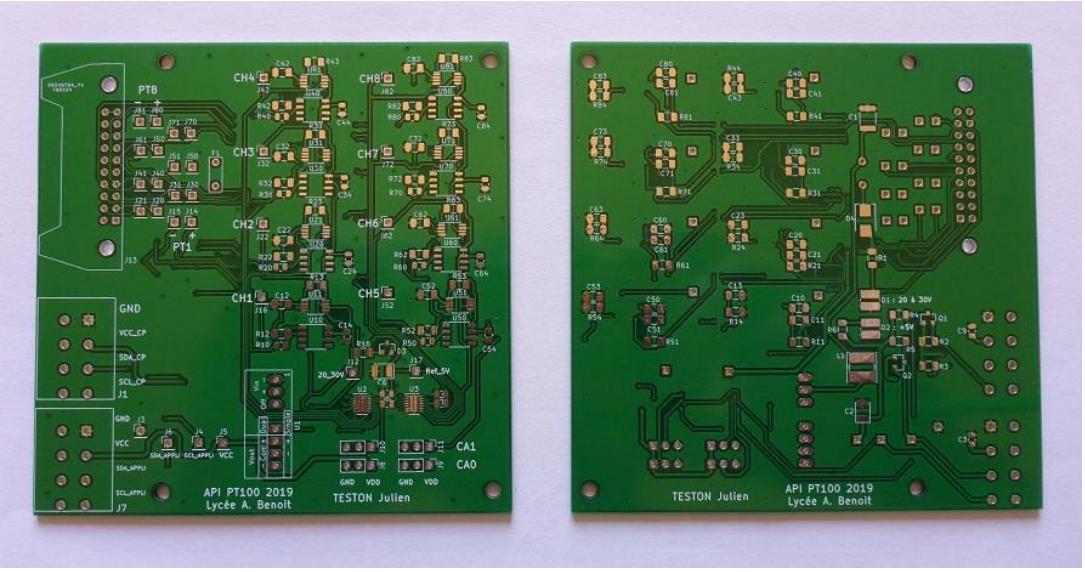


7.11. Fabrication des cartes

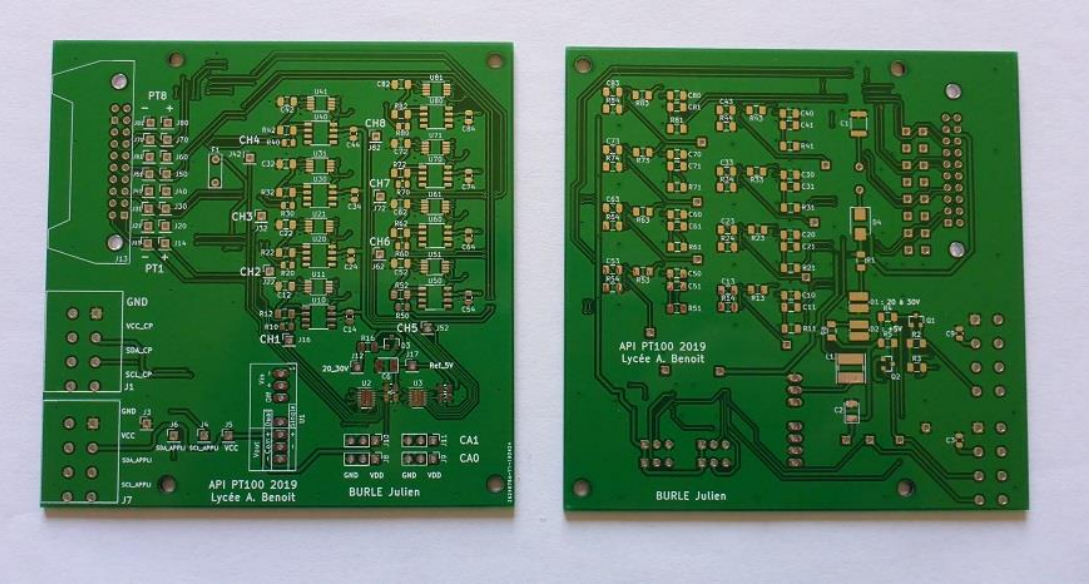
Teston Julien et moi avons créés nos propres cartes, elles ont donc des routages différents, mais ne sont pas pour autant si différentes

Les cartes ont été créées par JLCPCB

Cartes de Teston Julien :



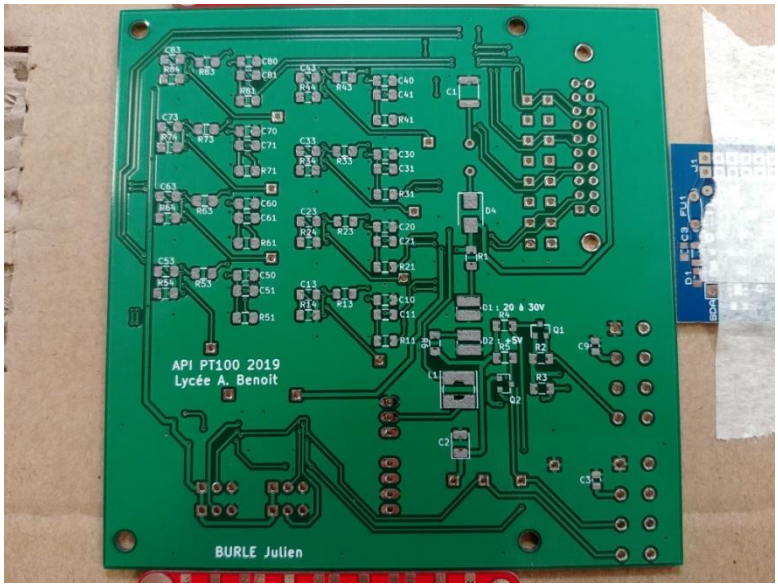
Mes cartes :



En utilisant un stencil fait par JLCPCB pour nos cartes, j'ai appliqué de la pâte à braser sur les empreintes des cartes

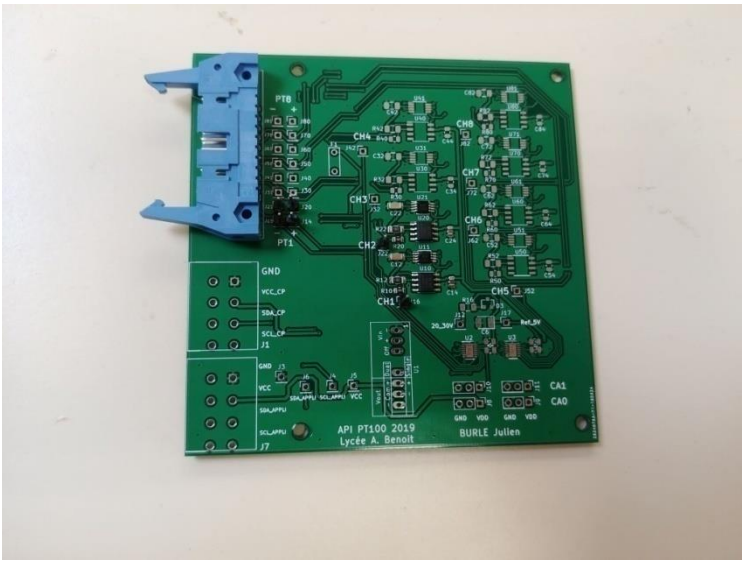
Notons aussi que Teston Julien aussi un stencil propre à ses cartes

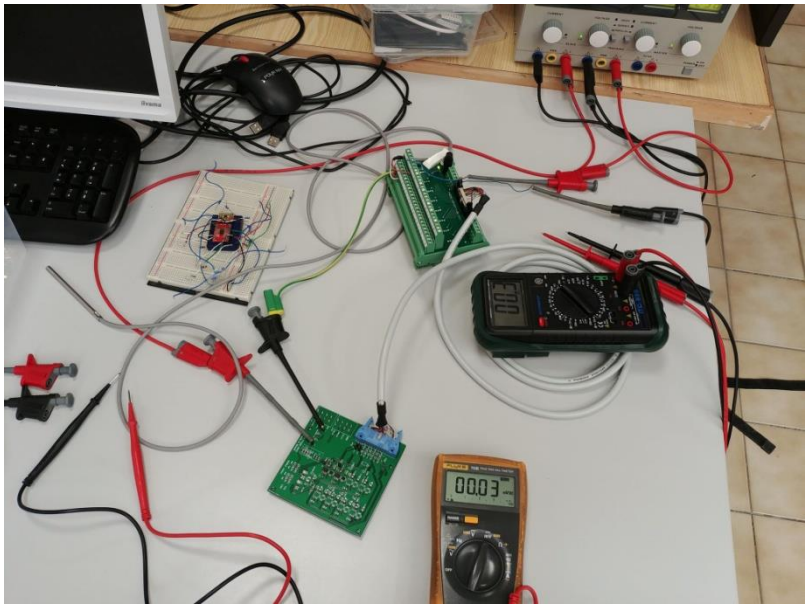
On peut voir ici la pâte à braser, et grâce à un microscope, on constate que cette pâte est composée de micro-billes, qui, une fois chauffées avec un four à refusion (ici le four FT02 de C.I.F.), souderont les composants aux empreintes



7.12. Tests de la carte

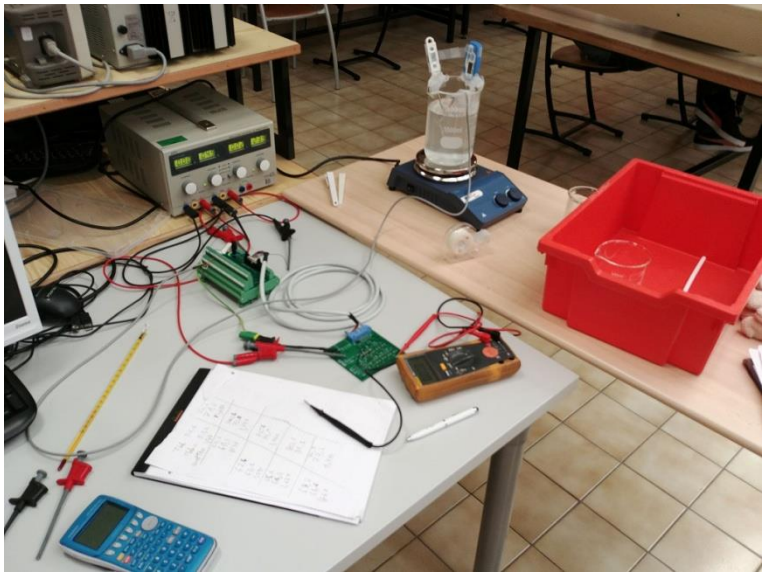
Pour tester ma carte et la solution, j'ai préparé une partie d'une carte et procédé à quelques tests simples, en prenant la température de la pièce avec la sonde PT100 et le module Phoenix





On peut voir ici le premier test de la carte, l'alimentation en haut à droite alimente le module Phoenix avec du 24V, et simule le convertisseur DC/DC TMR2411, qui changera la tension d'alimentation de 24V en 5V (qu'on peut voir au point VCC J5)

La carte partage aussi le point commun du module Phoenix



Nous avons ensuite utilisés du matériel de la classe de physique, on peut donc voir ici un bécher chauffé par un agitateur analogique chauffant RSLAB-1C

Grace à ce matériel, nous avons pu tester la sonde PT100 a des températures différentes de celle de la pièce, de 20 degrés à 80 degrés

Pour connaître la température des sondes PT100, il faut mesurer la tension entre les points +1 et -1 pour la sonde n°1, +2 et -2 pour la sonde n°2 etc., la tension à 25°C devrait être d'environ 11.00mV, le courant de référence est de 100 micro Ampères, nous avons alors a calculer U / I = R, donc 11mV / 0.0001A =110Ω, et en utilisant la table des températures de la sonde PT100, on trouvera donc la température mesurée

La tension de sortie mesurée a CH1, CH2 etc. peut être calculée grâce à la tension entre les point + et – de la sonde, la tension de la résistance de 78.7Ω et l'amplification de l'INA326

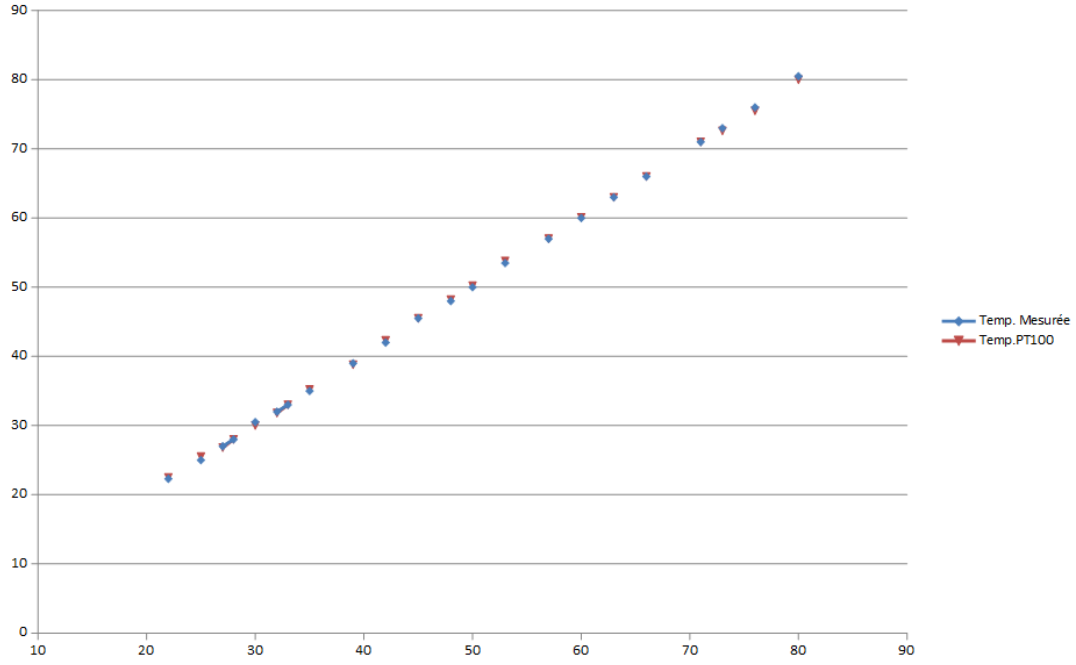
Par exemple, on mesure une tension d'11mV entre les points + et – de la sonde, la tension de la résistance de 78.7Ω est égale à 7.87mV car U = R*I, donc 78.7Ω * 0.0001A = 0.00787V

On soustrait ensuite la tension de la résistance de 78.7Ω à la tension entre les point + et – de la sonde, donc 11mV – 7.87mV = 3.13mV

Cette tension est ensuite multipliée par l'amplification de l'INA326, qui varie entre 348 et 350

0.00313 * 348 = 1.089V

On peut voir ici la mesure consistante de la sonde PT100 lors du test avec le bécher :



On constate que les mesures sont assez précises, et que la sonde PT100 a une marge d'erreur de moins d'1%

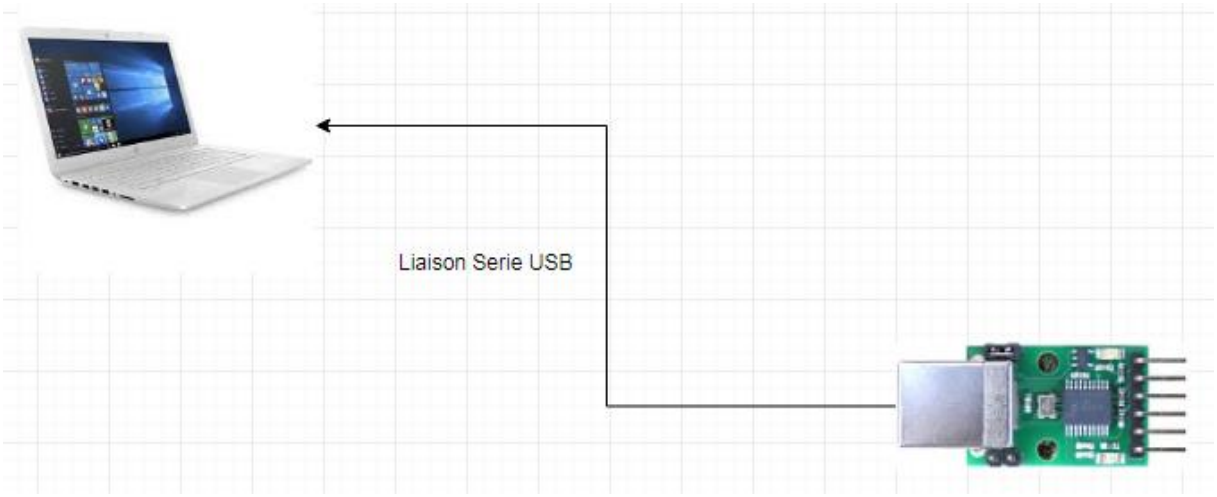
L'amplification ici varie entre 348 et 351, ce qui est correct

7.13. Conclusion

Ce projet m'a permis de travailler sur ma présentation, et m'a aussi donné de l'expérience en prenant compte d'anciens dossiers, faire des tests, créer des schémas et routages, et enfin fabriquer des cartes électroniques

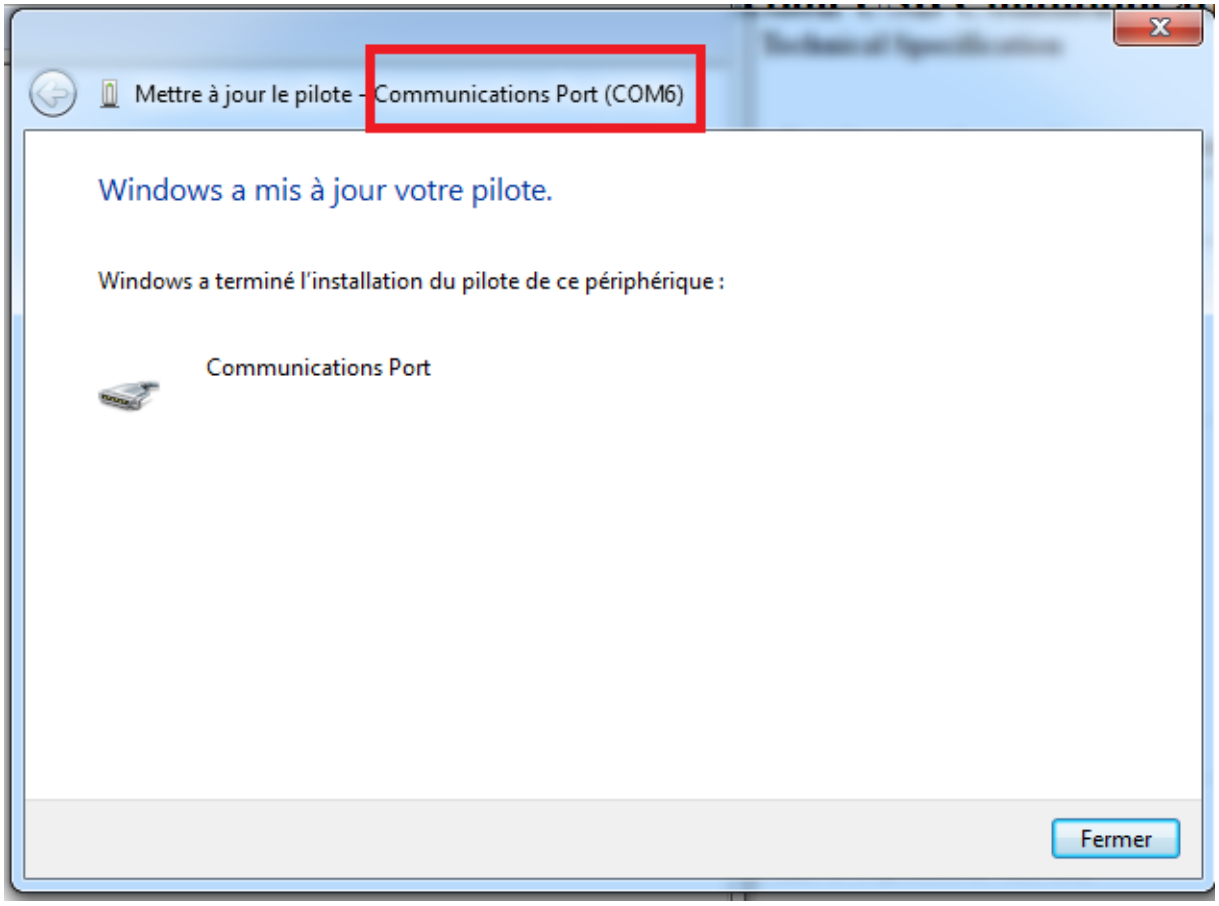
Ce projet m'a donc donné des connaissances nécessaires à mes études et ma future carrière professionnelle

L'entreprise API recevra nos résultats, et considèreront nos solutions, ce qui était le but de ce projet



	USB-I2C	USB-ISS	
Alimenté par USB	✓	✓	
Soutenir I2C à 100khz	✓	✓	
Soutenez I2C à 20khz-1000khz	✗	✓	Fixe 20,50,100,400 & 1000khz
I2C + I / O	✗	✓	
Mode SPI	✗	✓	24khz à 3Mhz
Mode série	✗	✓	300baud à 3Mbaud
I2C Direct	✗	✓	Utilisé pour créer des séquences I2C personnalisées
Bootloader pour les mises à jour utilisateur	✗	✓	Utilise le chargeur de démarrage Microchip
Alimenter le circuit externe	✓	✓	Jusqu'à 80 mA pour USB-ISS
Opération 5v	✓	✓	
Opération 3.3v	✗	✓	
Trous de montage sur le circuit imprimé	✗	✓	

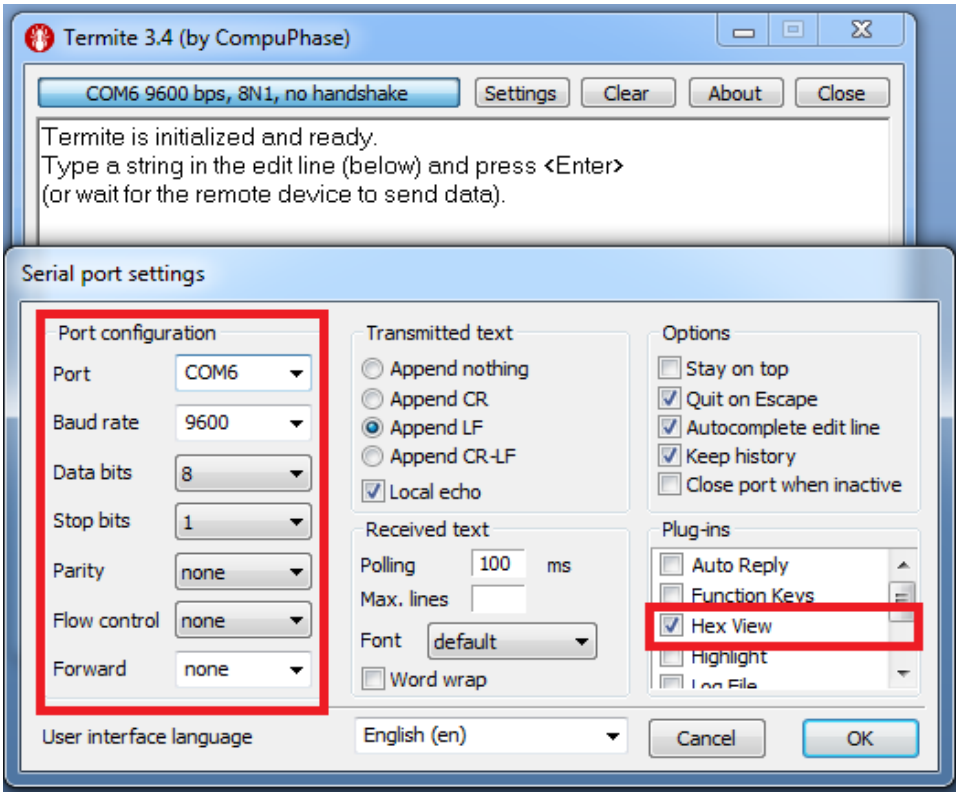
Avant de connecter le module USB-ISS, il a fallu télécharger le pilote et l'installer. L'USB-ISS apparait maintenant comme un port COM.



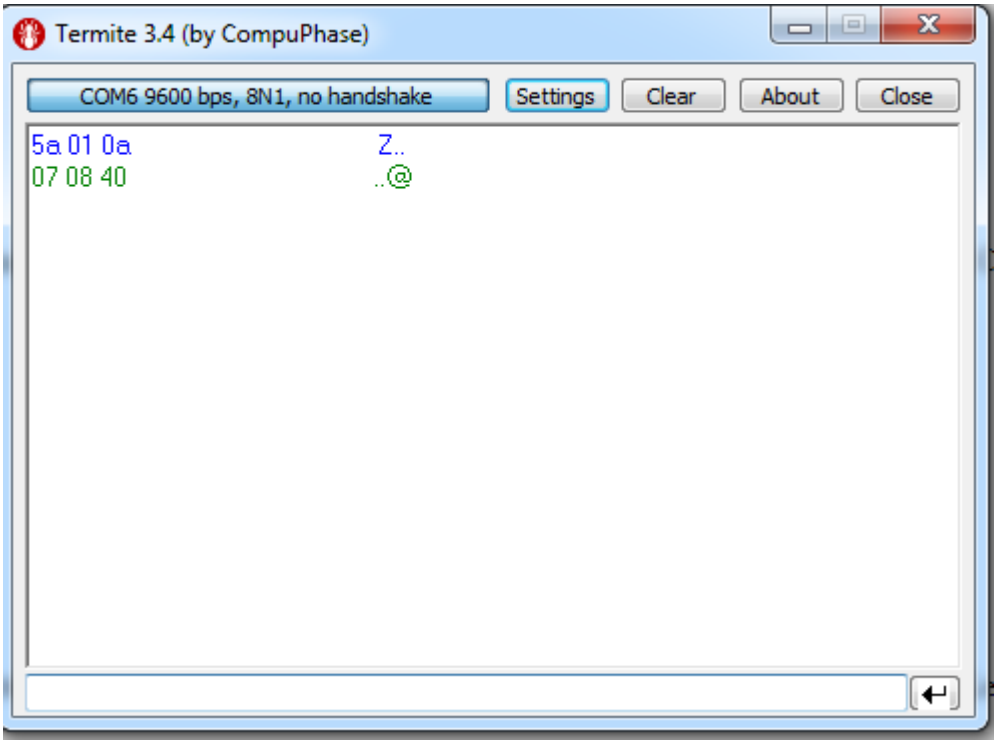
Commander	Sous-commande	La description
USB-ISS (0x5A)	ISS_VERSION (0x01)	Renvoie 3 octets, l'ID du module (7), la version du microprogramme et le mode de fonctionnement actuel.
USB-ISS (0x5A)	ISS_MODE (0x02)	Définit le mode de fonctionnement, I2C / SPI / Série, etc. Voir la section suivante pour plus de détails.
USB-ISS (0x5A)	GET_SER_NUM (0x03)	Renvoie le numéro de série USB unique de 8 octets du module.

Mode de fonctionnement	Valeur
IO_MODE	0x00
IO_CHANGE	0x10
I2C_S_20KHZ	0x20
I2C_S_50KHZ	0x30
I2C_S_100KHZ	0x40
I2C_S_400KHZ	0x50
I2C_H_100KHZ	0x60
I2C_H_400KHZ	0x70
I2C_H_1000KHZ	0x80
SPI_MODE	0x90
EN SÉRIE	0x01

Pour pouvoir communiquer avec l'adaptateur USB-ISS, j'utilise un logiciel nommé « termite » qui me permet de pouvoir communiquer en liaison série et de pouvoir lire/envoyer les données en hexadécimal.



Je débute par demander la version du firmware ISS :
ISS-VERSION renverra trois octets. Le premier est l'ID de module, il sera toujours égal à 7. Le deuxième octet est le numéro de révision du microprogramme. Le troisième octet est le mode de fonctionnement actuel, ISS_MODE. Ceci est initialisé à 0x40 (I2C-S_100KHZ) à la mise sous tension.



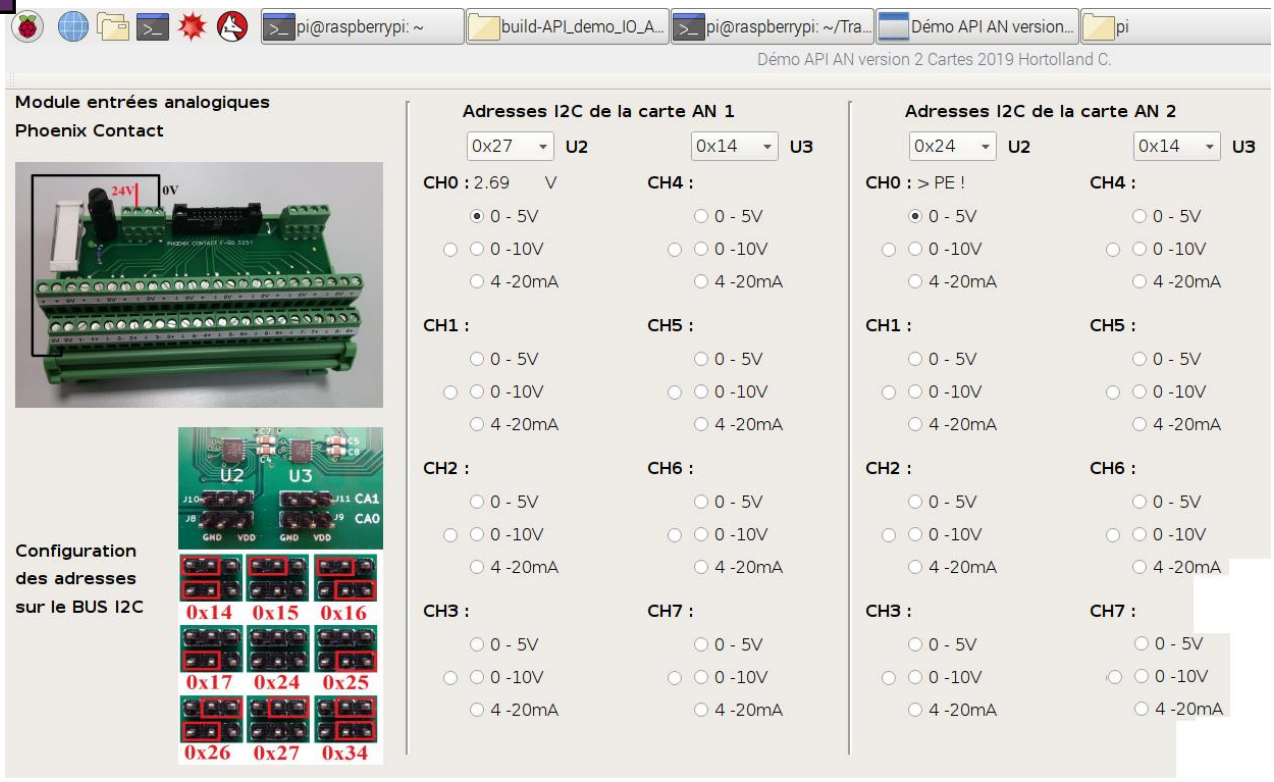
Le mode I2C-S_100KHZ n'est pas forcément le meilleur pour nous, le mode I2C-S_400KHZ est préféré

```
5a 02 50 0a                Z.P.
ff 00                      ÿ.

5a 01 0a                   Z..
07 08 50                   ..P
```

On voit ainsi que le mode à bien changé

Une fois l'adaptateur bien pris en main, on a enfin pu passer à la partie programmation

Présentation des applicationsQtCreator



Application disponible crée par Mr Hortolland

Présentations de quelques partie du code :

CarteAPIAN.h

```
#pragma once

class carteAPIAN
{
char adresseLTC2487_U
char adresseLTC2487_U
bool CH0__5V;
bool CH1__5V;
bool CH2__5V;
bool CH3__5V;
bool CH4__5V;
bool CH5__5V;
bool CH6__5V;
bool CH7__5V;

bool CH0__10V;
bool CH1__10V;
bool CH2__10V;
bool CH3__10V;
bool CH4__10V;
bool CH5__10V;
bool CH6__10V;
bool CH7__10V;

bool CH0__4_20mA;
bool CH1__4_20mA;
bool CH2__4_20mA;
bool CH3__4_20mA;
bool CH4__4_20mA;
bool CH5__4_20mA;
bool CH6__4_20mA;
bool CH7__4_20mA;

```

Déclaration des booléens pour les boutons radio de l'IHM.

MainWindow.cpp

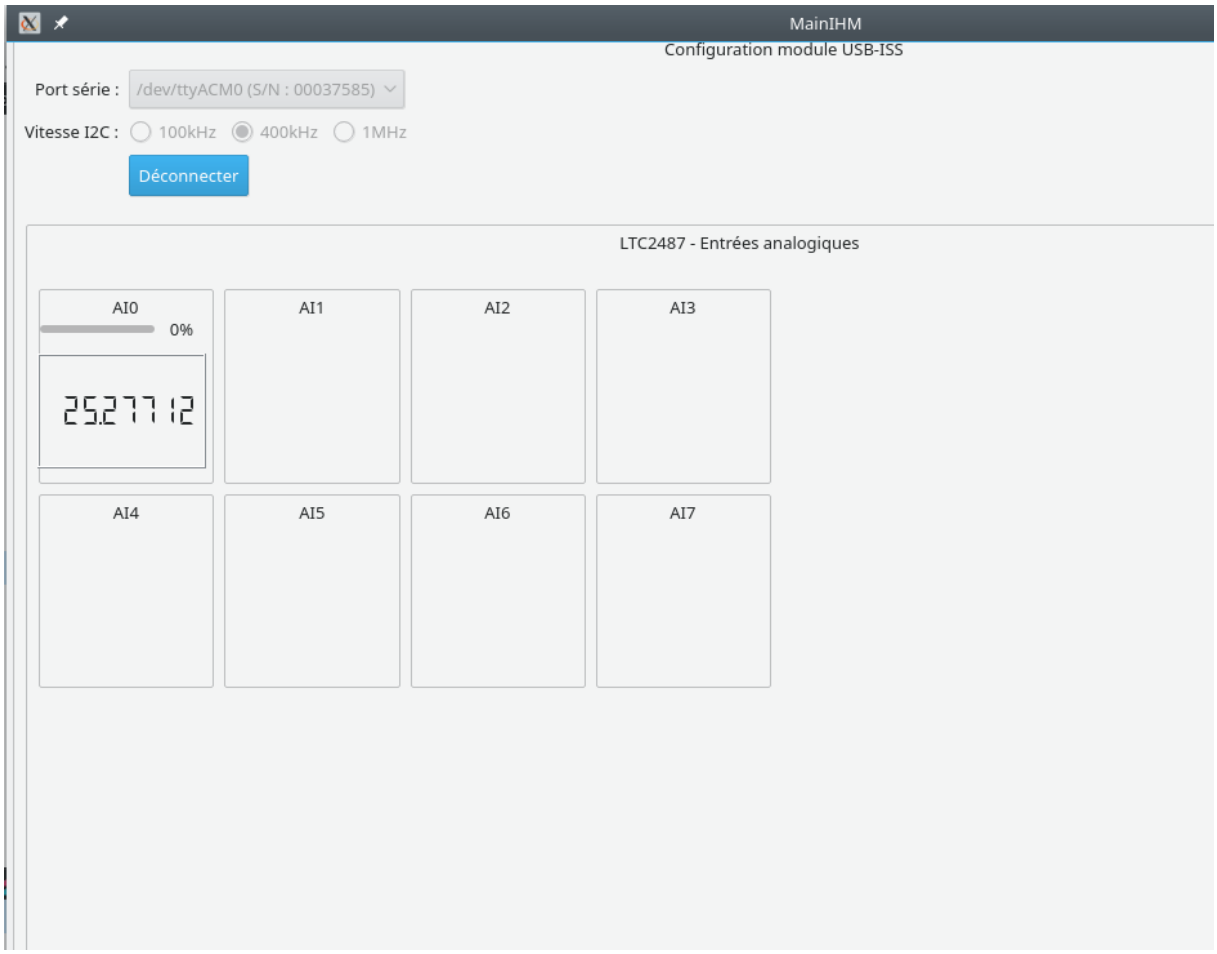
```
1   #include "mainwindow.h"
2   #include "ui_mainwindow.h"
3   #include <bcm2835.h>
4   #include "carteAPIIO.h"
5   #include "carteAPIAN.h"
6
7   #define clk_div BCM2835_I2C_CLOCK_DIVIDER_2500
8
9
10
11  MainWindow::MainWindow(QWidget *parent) :
12      QMainWindow(parent),
13      ui(new Ui::MainWindow)
14  {
15      ui->setupUi(this);
16      carteAN1 = new carteAPIAN(); // Instanciation d'une carte API AN
17      carteAN2 = new carteAPIAN(); // Instanciation d'une seconde carte API AN
18
19      // Paramétrage ComboBox AN
20      ui->comboBox_2->addItem("0x14");
21      ui->comboBox_2->addItem("0x15");
22      ui->comboBox_2->addItem("0x16");
23      ui->comboBox_2->addItem("0x17");
24      ui->comboBox_2->addItem("0x24");
25      ui->comboBox_2->addItem("0x25");
26      ui->comboBox_2->addItem("0x26");
27      ui->comboBox_2->addItem("0x27");
28      ui->comboBox_2->addItem("0x34");
29
30      ui->comboBox_3->addItem("0x14");
31      ui->comboBox_3->addItem("0x15");
32      ui->comboBox_3->addItem("0x16");
33      ui->comboBox_3->addItem("0x17");
34      ui->comboBox_3->addItem("0x24");
35      ui->comboBox_3->addItem("0x25");
36      ui->comboBox_3->addItem("0x26");
37      ui->comboBox_3->addItem("0x27");
38      ui->comboBox_3->addItem("0x34");
39
40      ui->comboBox_4->addItem("0x14");
41      ui->comboBox_4->addItem("0x15");
42      ui->comboBox_4->addItem("0x16");
43      ui->comboBox_4->addItem("0x17");
44      ui->comboBox_4->addItem("0x24");
45      ui->comboBox_4->addItem("0x25");
46      ui->comboBox_4->addItem("0x26");
47      ui->comboBox_4->addItem("0x27");
48      ui->comboBox_4->addItem("0x34");
49
```

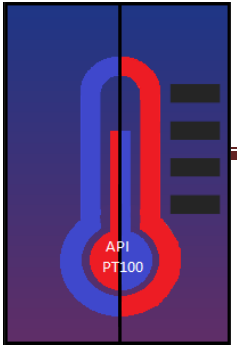
Ligne 20 à 48 : déclaration des adresses i2c pour l'IHM

```
void MainWindow::sltOnTimer()
{

    char i2cOut[3];
    char i2cIn[4];
    unsigned int valeur;
    double valeur_tension;
    double valeur_courant;

    // Récupération de l'adresse I2C pour la carte AN1 U2
    bcm2835_i2c_setSlaveAddress(carteAN1->getAdresseU2());

    // Gestion CH0 Carte AN1

    if (carteAN1->CH0_5V())
        {
            i2cOut[0]=0xB0;                 // Valide ADC, mode single, ch0
            i2cOut[1]=0x88;                 // Entrée externe, gain 1, speed2
            bcm2835_i2c_write(i2cOut, 2);   // Ecriture dans l'ADC
            bcm2835_delay(100);             // Tempo supérieure à celle de la conversion
            bcm2835_i2c_read(i2cIn, 3);     // Lecture des 3 octets du résultat
            if (i2cIn[0]>191)
                {
                this->ui->label_33->setText("> PE !");  // Tension supérieure à pleine échelle
                this->ui->label_34->setText(" ");       // Effacement de l'unité
                }
            else if (i2cIn[0]<128)
                {
                this->ui->label_33->setText("0V aj"); // 0V ajusté si tension très légèrement négative
                this->ui->label_34->setText(" ");      // Effacement de l'unité
                }
            else
                {
                valeur=((i2cIn[2]>>6) + (4*i2cIn[1]) + 1024*(i2cIn[0]&0b00111111)); // Mise en forme et calcul de la valeur reçue
                valeur_tension=((double)valeur*5)/65535;    // Calcul de la tension équivalente avec amplification de 2
                QString str = QString::number(valeur_tension,'g',3);
                this->ui->label_33->setText(str);
                this->ui->label_34->setText("V");
                }
        }
```

Lire les commentaires

Application QtCreator Cr é





Presentation de quelques parties de code :

MainIHM.h :

```
1  #ifndef MAINIHM_H
2  #define MAINIHM_H
3
4  #include <QDialog>
5  #include <QLabel>
6  #include <QTimer>
7
8  #include "usbissinfo.h"
9  #include "usbiss.h"
10 #include "ltc2487.h"
11
12 namespace Ui {
13 class MainIHM;
14 }
15
16 class MainIHM : public QDialog
17 {
18     Q_OBJECT
19
20 public:
21     explicit MainIHM(QWidget *parent = 0);
22     ~MainIHM();
23
24 private:
25     Ui::MainIHM *ui;
26     QList<UsbIssInfo> m_modules;
27     UsbIss * m_usbiss;
28     LTC2487 * m_ltc;
29     QTimer m_tick;
30     const uint8_t LTC2487_I2C_ADDRESS = 0x4E;
31
32 private slots :
33     void onBtnConnectClicked();
34     void onOutputChanged(bool state);
35     void onTimeOut();
36 };
37
38 #endif // MAINIHM_H
39
```

Ligne 30 : l'adresse I2C etant sur 7 bits, mais Qt envoi 8 bits, il faut donc decaler l'adresse en rajoutant un bit de poid faible, donc pour l'adresse 27 (010 0111) se transforme en 4E (0100 1110).

usbISS.h

```
2   #define USBISS_H
3
4   #include <QObject>
5   #include <QSerialPort>
6
7   #include "usbissinfo.h"
8
9   class UsbIss : public QObject
10  {
11      Q_OBJECT
12  public:
13      enum I2cSpeed {
14          sw_100kHz = 0x40,
15          sw_400kHz = 0x50,
16          hw_100kHz = 0x60,
17          hw_400kHz = 0x70,
18          hw_1MHz = 0x80,
19      };
20      Q_ENUM(I2cSpeed)
21
22      explicit UsbIss(UsbIssInfo& info, I2cSpeed scl, QObject *parent = nullptr);
23
24      QByteArray readAD1(uint8_t deviceAddr, uint8_t regAddr, int nbBytesToRead, bool& status);
25      uint8_t readAD1(uint8_t deviceAddr, uint8_t regAddr, bool& status);
26      void writeAD1(uint8_t deviceAddr, uint8_t regAddr, QByteArray data, bool& status);
27      void writeAD1(uint8_t deviceAddr, uint8_t regAddr, uint8_t data, bool& status);
28
29      QByteArray readAD0(uint8_t deviceAddr, int nbBytesToRead, bool& status);
30      uint8_t readAD0(uint8_t deviceAddr, bool& status);
31      void writeAD0(uint8_t deviceAddr, QByteArray& data, bool& status);
32      void writeAD0(uint8_t deviceAddr, uint8_t data, bool& status);
33
```

Ligne 13 à 18 :

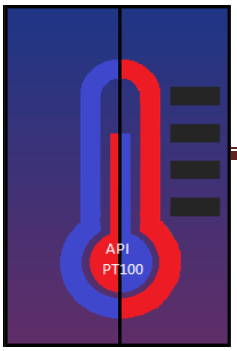
Declaration de la vitesse I2C software ou hardware.

LTC2487.cpp

```cpp
#include <QThread>
#include <QDebug>
#include "ltc2487.h"


LTC2487::LTC2487(UsbIss& usbiss, uint8_t deviceAddress, QObject *parent) : QObject(parent)
    , m_usbiss(usbiss), m_deviceAddr(deviceAddress)
{

}

double LTC2487::getAnalog(int channel) {
    static uint8_t muxAddress[] = {0xb0, 0xb8, 0xb1, 0xb9};
    QByteArray request;
    QByteArray response;
    bool ok;
    uint16_t ana;
    uint16_t ana2;
    request.append(muxAddress[ channel ]);
    request.append(0x88);

    m_usbiss.writeAD0(m_deviceAddr, request, ok);

    QThread::usleep(100000);

    response = m_usbiss.readAD0(m_deviceAddr, 3, ok);

        qDebug() << "valeur convertisseur : " + QString(response.toHex());

    //  xx00 0000 0000 0000 00xx xxxx
    ana = (response.at(2) >> 6) & 0x03;
    ana += response.at(1) * 4;
    ana += (response.at(0) & 0x3F) * 1024;


    return  ((((((ana/65535.0 * 5.0)/2)/355)+78.7*0.0001) / 0.0001) /100 -1)/0.00385;
                valeur en Volt
                                valeur en Degres Celsius


}
```

Ligne 31 à 33 : recuperation de la donnée du LTC2487 sur 3 octet

Ligne 36 : Conversion de la donnée analogique en numerique, puis en degres celsius



Presentation partie du code emulateur LTC2487 avec une carte Arduino

```
#include <Wire.h>

const uint8_t LTC_I2C_ADDR = 0x24; // adresse 7 bits du composant lorsque les broches CA0 et CA1 sont forcées à 0
uint8_t analogPin = 0; // n° de la broche An (0=A0,...3=A3) à lire lors de la prochaine requête de lecture
char msg[64];

void setup() {
  // Config I2C à 400kHz puis ouverture
  Wire.setClock(400000);
  Wire.begin(LTC_I2C_ADDR);

  // Mise en place des gestionnaires d'évènement
  Wire.onRequest(readRequestFromMasterHandler);
  Wire.onReceive(writeRequestFromMasterHandler);

  // Ouverture liaison série pour debug
  Serial.begin(9600);
  Serial.println("** Emulateur LTC **");
}
```

On commence par choisir l'adresse du bus I2C, et le numero de la broche sur la carte Arduino.

On initialise ensuite la configuration I2C à 400kHz et debute la communication sur l'adresse I2C choisie.

On met en place les gestionnaire d'evenement qui seront utilisé lorque le maitre veut lire ou ecrire sur le LTC2487.

Et on initialise enfin la liaison série à 9600Bauds.

```
void readRequestFromMasterHandler() {
  uint8_t ltcOut[3]; // 3 octets qui représentent les 24 bits retournés par le LTC2487

  // On lit la valeur analogique sur la voie pré-sélectionnée et on effectue une mise à l'échelle
  // (0...1023 -> 0...65535) 10Bits à 16 bits
  unsigned short vIn = map(analogRead(analogPin), 0, 1023, 0, 65535);

  // On formate la valeur en accord avec celle que renvoit un LT2487 réel
  ltcOut[2] = (vIn & 0x03) << 6;
  ltcOut[1] = (vIn >> 2) & 0xFF;
  ltcOut[0] = (vIn >> 10) & 0xFF;

  // On envoir la valeur formatée sur le bus
  Wire.write(ltcOut, 3);

  // Message d'information pour s'assurer de la prise en compte de la commande
  sprintf(msg, "< Valeur voie %d : 0x%02x%02x%02x (24bits) /  0x%04x (16bits)", analogPin, ltcOut[0], ltcOut[1], ltcOut[2], vIn);
  Serial.println(msg);
```

Declaration du gestionnaire d'evenement pour ecrire sur le LTC2487 :

Declaration d'un tableau a 3 octets qui represente les 24 bits que renvoi le LTC2487.

On lit ensuite la valeur analogique sur la broche et on effectue une mise a l'echelle, on passe de 10 bits a 16bits pour etre en accord avec la valeur que renvoi reellement le LTC2487.

On stocke les valeurs dans le tableau qu'on envoie ensuite.

Mise en place d'un message d'information pour s'assurer que la commande à bien était prise en compte.